A Comparative Evaluation of Population-based Optimization Algorithms for Workflow Scheduling in Cloud-Fog Environments

by

Dineshan Subramoney

UNIVERSITY *of the* WESTERN CAPE

This thesis is submitted in fulfillment of the academic requirements

for the degree of

Master of Science in Computer Science

In the Department of Computer Science

Faculty of Natural Sciences

December 15, 2022

ABSTRACT

Scientific workflows are denoted by interdependent tasks and computations that are aimed at achieving some scientific objectives. The scheduling of these workflows involve the allocation of the tasks to particular computational resources, traditionally on the cloud infrastructure. This process is, however, very challenging. It is associated with high computation and communication costs because scientific workflows are data-intensive and computationally complex.

In recent years, there has been overwhelming interest in using population-based optimization algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) for scientific workflow scheduling, predominantly, in the cloud environments. Cloud infrastructure provides cost efficiency, high speed, excellent accessibility, elasticity, virtualization capabilities and sporadic batch processing. Pegasus workflows, such as Montage, Cybershake, Epigenomics, LIGO and SIPHT, are used in all these studies. Makespan and cost are used as the optimization objectives, while other equally important objectives, such energy consumption and load balancing, are not usually addressed. Furthermore, the canonical PSO, which is popularly used in most of these works suffer from premature convergence.

As demand grows, the cloud computing paradigm faces challenges such as high latency which leads to low quality of service (QoS). Fog computing, a decentralized computing infrastructure, located between the end devices and the cloud, has been proposed as the solution to address the limitations of cloud infrastructure. This emerging three-tier network, that incorporates end-devices, fog devices and cloud devices, is yet to be fully utilized for scientific workflow scheduling. Therefore this work adopts this futuristic network.

To deal with aforementioned problems, the first phase of this work proposes a weighted sum objective function which incorporates four objectives: makespan, cost, energy and load balancing for

cloud and fog tiers. A comparative performance evaluation of PSO, GA, GA-PSO and DE, which has been used for scientific workflow scheduling for the first time in this work, is then performed. The FogWorkflowSim Toolkit is used for the evaluation process, with the objectives serving as performance metrics. The GA-PSO algorithm exhibits better performance compared to the other approaches.

In the second phase, the Multi-Swarm Particle Swarm Optimization (MS-PSO) algorithm is proposed in order to deal with PSO's problem of premature convergence. Simulation results show that the proposed Multi-Swarm based PSO algorithm outperforms the canonical PSO by at least 10% on all scientific workflows. It also competes fairly well against the other approaches and it is more stable and reliable. The Multi-Swarm based PSO only ranks second to the canonical PSO, in terms of execution time.

In the third phase, a more modern Differential Evolution (DE) variant known as SHADE is applied to workflow scheduling and it is compared with the canonical DE. The SHADE variant outperformed the canonical DE by at least 10%, for each performance metric, on all scientific workflow instances.

In future, multiple species, incorporating population update mechanisms from several algorithmic frameworks (MS-PSO, DE, GA), will be studied. Hybdridization of the algorithms proposed in this work with dynamic approaches and multi-objective reinforcement learning-based techniques will be also investigated.

Keywords: Scientific Workflows, PSO, Genetic Algorithms, Differential Evolution, Fog Computing, Cloud Computing

PUBLICATIONS

This work has led to the following publications.

Journals

1. **D. Subramoney** and C. N. Nyirenda, "Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments," in *IEEE Access*, vol. 10, pp. 117199-117214, 2022. (Published).

Conferences

1. **D. Subramoney** and C. Nyirenda, "Success-History-based Differential Evolution (SHADE) for Scientific Workflow Scheduling in Cloud-Fog Environments," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2022*, 2022, pp. 132–137. (Published).

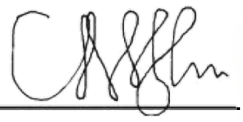
2. **D. Subramoney** and C. Nyirenda, "PSO-based workflow scheduling: A comparative evaluation of cloud and cloud-fog environments," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021*, 2021, pp. 258–263. (Published).

3. **D. Subramoney** and C. N. Nyirenda, "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 760–767. (Published).

CERTIFICATION

As the candidate's supervisor, I have approved this thesis for submission.

Supervisor: Dr. C. N. Nyirenda

Signature: 

Date: 15/12/2022

DECLARATION

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This thesis is being submitted for the degree of Master of Science in Computer Science at the University of the Western Cape. It has not been submitted to any other university for any other degree or examination.

Dineshan Subramoney

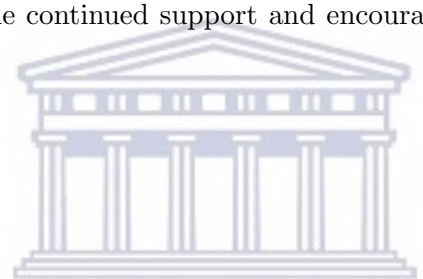
15/12/2022

Date

ACKNOWLEDGEMENT

I would like to express my appreciation to Dr Clement Nyirenda, my supervisor, for his guidance and motivation throughout this research. Thank you for the constructive criticism and weekly meetings, and your patience as I juggled work commitments with my studies.

To my family, thank you for the continued support and encouragement throughout my academic endeavours.



Contents

List of Acronyms

DAG	Direct Acyclic Graph
DE	Differential Evolution
DVFS	Dynamic Voltage and Frequency Scaling
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
GSS	Granularity Score Scheduling
HEFT	Heterogeneous Earliest Finish Time
HPC	High Performance Computing
IoT	Internet of Things
IPAC	Infrared Processing and Analysis Center
IWD	Intelligent Water Drops
LAGA	Look-Ahead Genetic Algorithm
LIGO	Laser Interferometer Gravitational Wave Observatory
Mbps	Megabits per Second
MIPS	Million Instructions Per Second
mW	Milliwatt
NASA	National Aeronautics and Space Administration
NCBI	National Center for Biotechnology Information
PETS	Performance Effective Task Scheduling
PSHA	Probabilistic Seismic Hazard Analysis
PSO	Particle Swarm Optimization
QoS	Quality of Service
RD	Reliability-Driven

SCEC	Southern California Earthquake Center
SHADE	Success-History-based Adaptive Differential Evolution
VM	Virtual Machine
XML	eXtensible Markup Language

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background and Orientation of the Study

Over the recent decades, there has been a rapid adoption of cloud computing services in various industries due to the value found in producing, storing and processing of massive amounts of data [2, 3]. Cloud computing provides computing resources such as compute, storage and networking services on-demand over the internet [4, 5]. Some of the benefits of cloud computing are cost efficiency, high speed, excellent accessibility, elasticity, virtualization capabilities and sporadic batch processing. These characteristics have made the cloud computing paradigm the ideal choice for implementing scientific workflows [6, 7]. Historically, the execution of scientific workflows were performed on High Performance Computing (HPC) [8] and distributed systems [9, 10].

Scientific workflows are denoted by interdependent tasks and computations that are used in various scientific domains. However, scheduling of scientific workflows in a distributed environment is a challenging problem and is considered NP-complete. Due to the data-intensive and computationally complex nature of scientific workflows, there are high computation and communication costs involved with the scheduling of these workflows, which entails the allocation of interdependent tasks in the workflow to the available virtual machines in the cloud [11]. Therefore, various optimization techniques have been proposed for efficient scheduling of tasks or workflows on the cloud infrastructure. Over the years, there has been a rapidly growing interest in the use of population-based algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) in the scheduling of scientific workflows in the cloud [12, 13, 14, 15].

However, as the demand for cloud services grow, the classic cloud computing paradigm faces several challenges such as high latency and low quality of service (QoS). Fog computing has been proposed as the solution to address these limitations [2, 16]. The fog computing paradigm is an extension of the cloud computing model and acts as an intermediate layer bringing limited capabilities of processing, storage and networking services closer to the end devices in a distributed manner. Any device with computing, storage and networking capabilities can be considered a fog device [17, 16]. Fog devices have the following advantages over their cloud counterparts: lower latency, improved user experience, higher security, and energy efficiency. The purpose of the fog layer is to improve efficiency, performance and reduce the amount of data transferred to the cloud for processing and storage. Therefore, some workflow tasks can be sent to fog devices for processing, instead of transferring them to the cloud, thus reducing network traffic and latency [18].

The cloud-fog computing paradigm provides a great platform for applications that are latency-sensitive and are characterized by complex data and long sessions of distributed computing such as the execution of scientific workflows. Hence, the application and development of optimization algorithms for efficient workflow scheduling on the emerging three-tier cloud-fog architecture, while meeting various QoS objectives, is an important challenge and it is a largely unexplored area of research.

1.2 Problem Statement

The research problem being addressed in this work is four-pronged, and is hereby presented as follows:

- Most studies on population-based optimization approaches for scientific workflow scheduling are based on the traditional two layered architecture, comprising the cloud servers and end devices [12, 13, 14, 15]; there are very few studies on the more progressive three-tier cloud-fog framework [12, 19].
- Most studies have focused only on makespan and cost, as the optimization objectives. Other equally important objectives such as energy consumption and load balancing among the computational elements are not usually incorporated in the algorithm.

- The canonical PSO, which has been used in many scientific workflow scheduling algorithms [20, 15], suffers from premature convergence [12, 21]. This creates sub-optimal solutions.
- After PSO, GA is the other popular population-based algorithm in scientific workflow scheduling; there is a dearth in literature on the use of Differential Evolution (DE) in this domain. This technique has, however, been equally successful in other application domains such as deep learning [22, 23], energy systems [24, 25], and robotics [26, 27]. This clearly highlights the enormous potential of DE.

1.3 Research Objectives

The main objective of this research is the development and comparison of population-based multi-objective algorithms for workflow scheduling in cloud-fog environments. The sub-objectives of this study are as follows:

1. To develop a weighted sum-based objective function incorporating four objectives for the optimization process: makespan, cost, energy consumption and load of a virtual machine (VM) on the respective cloud and fog layers.
2. To implement and compare several canonical population-based algorithms for scientific workflow scheduling in the three-layered architecture, incorporating fog and cloud servers.
3. To develop and implement Multi-Swarm PSO for scientific workflow scheduling and compare it with the canonical PSO.
4. To develop and implement the DE variant known as Success-History-based Adaptive Differential Evolution (SHADE) for scientific workflow scheduling and compare it with the canonical DE.
5. To evaluate the performance of the algorithms proposed in this study using the FogWorkflowSim simulator [28].

1.4 Significance of the Study

This study will demonstrate the effectiveness of different population-based algorithms for workflow scheduling in cloud-fog environments to enable faster and more efficient processing of scientific workflows, and quicker response times for critical use cases. Furthermore, the incorporation of crucial objectives such as energy consumption and load balancing in cloud-fog architectures will illustrate the importance of considering the environmental impact of running data centers, given the recent high adoption of multi-tier computing networks. This work will also open a new frontier of knowledge in the area of population-based multi-objective optimization and its application to workflow scheduling in cloud-fog environments.

1.5 Limitations

This study uses the simulator known as FogWorkflowSim [28] for the simulation of scientific workflow execution in a cloud-fog environment due to the high economic cost associated with setting up a real-life cloud-fog implementation. The simulations are repeated several times for purposes of credibility and to ensure reliability of the results.

1.6 Delimitations

The scientific workflows used in this case are only those from the Pegasus workflow management system [29]. These workflows are output files containing a respective workflow's task dependencies, run-times and task sizes which are based on real-life workflow executions. These workflows are chosen since they are well-known and have important real world scientific applications in their respective fields [1]. Furthermore, they have been widely used in practice for research endeavours.

1.7 Contributions

The main contribution of this work is the evaluation, comparison and efficacy of population-based optimization approaches to scientific workflow scheduling in a three-tier cloud-fog environment setup. Specifically, the contributions of this study can be presented as follows:

1. A weighted sum-based objective function is developed incorporating four crucial objectives: makespan, cost, energy consumption, and load balancing.
2. This work proposes a Multi-Swarm PSO-based algorithm for workflow scheduling with the aim of addressing the known issue of premature convergence with the canonical PSO algorithm.
3. A survey on recent state-of-the-art multi-swarm PSO-based approaches.
4. Introduction of the DE algorithm to the workflow scheduling problem.
5. A popular DE variant, known as SHADE, is implemented for scientific workflow scheduling and its performance is compared with the canonical DE.

These contributions have led to the following publications:

1. **D. Subramoney** and C. N. Nyirenda, "Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments," in *IEEE Access*, vol. 10, pp. 117199-117214, 2022.
2. **D. Subramoney** and C. Nyirenda, "Success-History-based Differential Evolution (SHADE) for Scientific Workflow Scheduling in Cloud-Fog Environments," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2022*, 2022, pp. 132–137.
3. **D. Subramoney** and C. Nyirenda, "PSO-based workflow scheduling: A comparative evaluation of cloud and cloud-fog environments," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021*, 2021, pp. 258–263.
4. **D. Subramoney** and C. N. Nyirenda, "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 760–767.

1.8 Thesis Organization

The rest of the thesis is organized as follows. **Chapter 2** presents the literature review on scientific workflows along with a discussion of population-based algorithms applied to the workflow scheduling problem. The cloud-fog paradigm is also reviewed. Finally, a brief overview is presented on the evolution of simulation frameworks used to conduct performance evaluations.

Chapter 3 presents the population-based algorithms for workflow scheduling. The construction of the weighted sum based objective function is also presented. Furthermore, the scheduling optimization process is discussed, followed by the simulations conducted and the results. **Chapter 4** proposes a Multi-Swarm PSO (MS-PSO) algorithm for workflow scheduling in cloud-fog environments. The expansion of the weighted sum objective function is described, followed by a review of the state-of-the-art Multi-Swarm algorithms. Finally, the performance evaluation and discussion of the results is presented.

Chapter 5 proposes the Success-History-based Differential Evolution (SHADE) approach to the scientific workflow scheduling problem in cloud-fog environments. The SHADE variant is described followed by the simulation environment. Finally, the simulation results along with a discussion is presented. **Chapter 6** concludes the research and discusses future perspectives.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents a comprehensive review related to this study and it is organized as follows: section 2.2 presents the basics of scientific workflows in literature. Section 2.3 discusses the Cloud-Fog environments while Section 2.4 presents population-based algorithms. Sections 2.5 presents population-based approaches to workflow scheduling. Section 2.6 discusses various simulation frameworks. Finally, Section 2.7 concludes the chapter.

2.2 Basics of Scientific Workflows

Workflow applications are generally modelled as a Direct Acyclic Graph (DAG) [12, 13, 14], defined by tuple $G = (T, E)$, where $T = \{t_1, t_2, \ldots, t_n\}$ denotes the set of n tasks and E is the set of edges, representing the temporal dependencies or precedence constraints between pairs of tasks in a workflow. An edge can be visualised by using inter-task data, $d_{ij} =< t_i, t_j >\in E$, where d_{ij} is a positive value representing the output data from task t_i which serves as input to task t_j. Therefore, the execution of task t_j will only start after the execution of task t_i has been completed. Task t_i is the parent task while task t_j is the child task. The very first task to start executing in a graph does not have a parent; it is known as an *entry task* t_{entry}. At the other end, the final task in a graph does not have any child and it is known as *exit task* t_{exit}. Fig 2.1 illustrates a sample structure of a workflow model. The tasks of the workflow placed horizontally on the same level can execute in

parallel order. In this example, tasks t_2, t_3 and t_4 can execute in a parallel order.

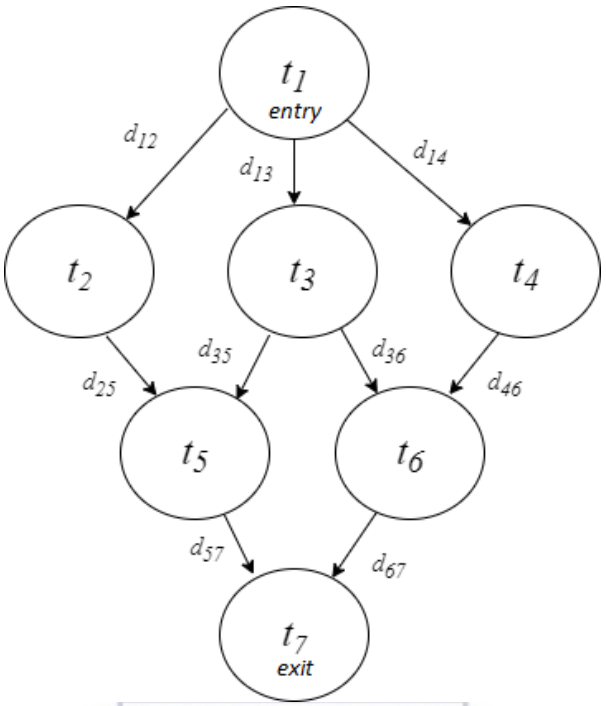


Figure 2.1: Sample workflow model with seven tasks

2.2.1 Components of workflows

The basic structures or components of workflows are shown in Fig 2.2. The final structure of a scientific workflow is usually composed of a combination of such components.

- *Process* - The simplest structure that operates on some input data to produce an output.
- *Pipeline* - It is composed of several data processing jobs combined sequentially. In this structure, each job in the pipeline processes the output of the previous stage and the output produced is used as input for the next stage in the pipeline.
- *Data distribution* - This may either produce output data that is used by multiple jobs or may operate on large datasets and divide them into smaller subsets to be processed by other jobs in the workflow. The partitioning leads to increased parallelism in the later stages of the workflow and is one of the main reasons for partitioning the data.
- *Data aggregation* - The jobs aggregate and processes the outputs of several individual jobs

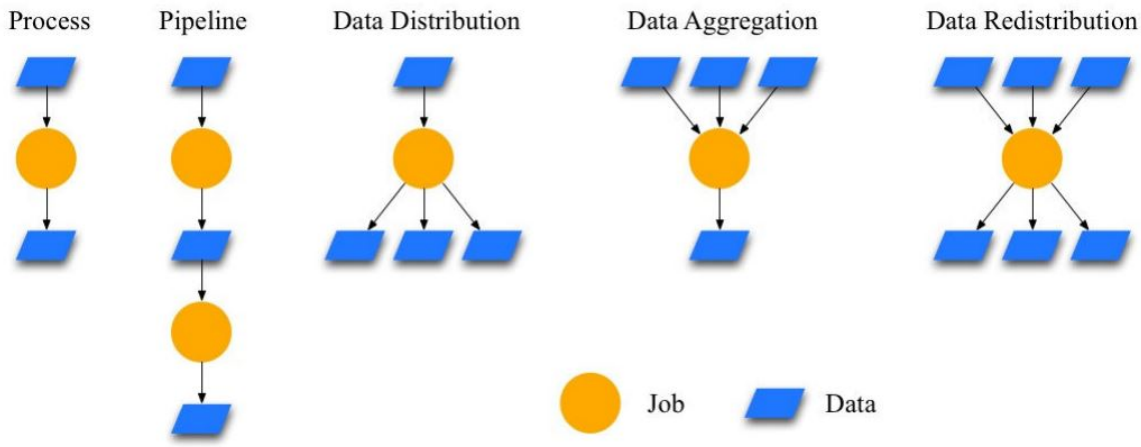


Figure 2.2: Basic workflow structures [1]

and generate a combined data product. As data aggregation jobs operate on several individual data inputs, it can potentially consume huge processing time. These jobs represent a reduction in the parallelism of the workflow.

- *Data redistribution* - There are instances where data aggregated from a previous stage are redistributed to multiple jobs in a following stage. The data redistribution jobs may be potential processing bottlenecks, however the parallelism is once again achieved in future stages.

2.2.2 Characterization of Scientific Workflows

This study uses five well-known scientific workflows [1] from various scientific domains for the effective evaluation of the proposed algorithms. These scientific workflows are published by the Pegasus project [29], where the Direct Acyclic Graph (DAG) is represented in an Extensible Markup Language (XML) file for the respective workflow. The workflows are composed of a number of tasks, dependencies, run-times and data required to be transferred between different tasks. Fig. 2.3 shows a simplified graphical structure of the workflows. These workflows are described as follows:

1. *Montage workflow*: Montage was created by the National Aeronautics and Space Administration (NASA)/Infrared Processing and Analysis Center (IPAC) Science Archive as an open source toolkit. This workflow is an astronomy application that creates custom mosaics of the

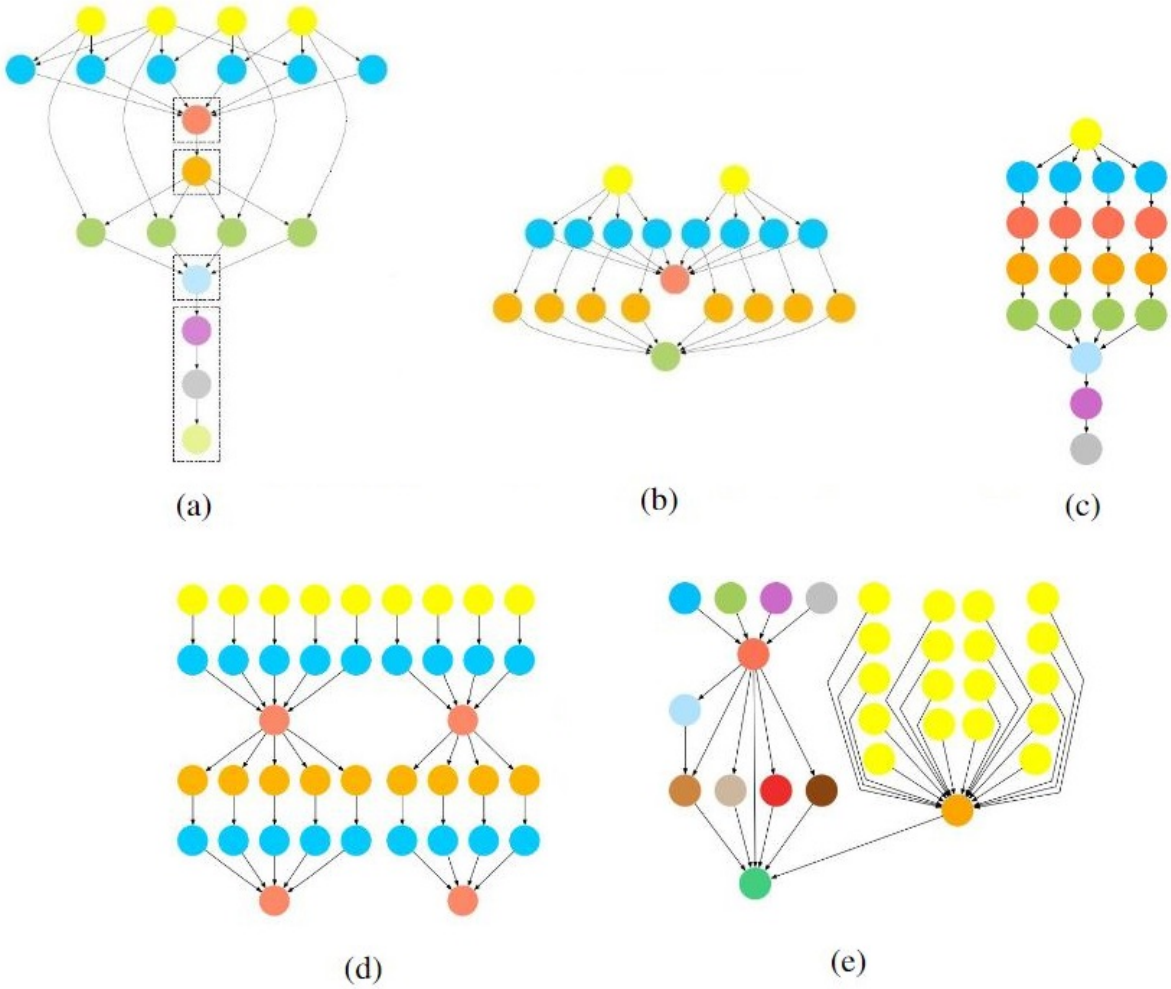


Figure 2.3: Structure of scientific workflows: (a) Montage, (b) CyberShake, (c) Epigenomics, (d) LIGO and (e) SIPHT.

sky with multiple input images.

2. *CyberShake workflow*: This is used by the Southern California Earthquake Center (SCEC) to characterize earthquake hazards threatening a region, using the Probabilistic Seismic Hazard Analysis (PSHA) technique.

3. *Epigenomics workflow*: This is used in the field of bioinformatics to automate the different operations in genome sequence processing. The workflow is also being used by the Epigenome Center in the processing of production DNA methylation and histone modification data.

4. *LIGO Inspiral Analysis workflow*: Is the Laser Interferometer Gravitational Wave Obser-

vatory (LIGO) for the detection of gravitational waves produced by various events in the universe as per Einstein's theory of general relativity. It is used to analyze the data obtained from the merging of compact binary systems such as binary neutron stars and black holes.

5. *Sipht workflow*: This is the bioinformatics project at Harvard University for conducting a wide search for small untranslated RNAs (sRNAs) that regulate several processes such as secretion or virulence in bacteria. This workflow automates the search for sRNA encoding-genes of all the bacterial replications in the National Center for Biotechnology Information (NCBI) database.

The next sub-section compares the cloud and fog computing paradigms.

2.3 Cloud-Fog Computing Environments

The classic cloud computing model offers many benefits to users in terms of improved reliability, reduced cost, elimination of hardware maintenance and administration, and a pay-as-you-use billing model [7]. Although these are notable advantages, cloud computing faces several challenges such as high latency, high bandwidth and potential network failures. The centralized and remote processing in cloud computing may also not be suitable for certain types of networks such as sensor networks and Internet of Things (IoT) [16]. These services are known as location-aware services and require location dependent processing with low latency. In order to overcome these challenges, a new computing model has been recently proposed, known as fog computing [17, 30, 31], where the capabilities of the cloud are located either at the network edge or very close to it. Table 2.1 summarizes the main differences between cloud and fog computing. It can be observed that cloud computing has limitations with respect to QoS demanded by real time applications requiring almost immediate response by the cloud servers.

In the three-layer architecture, specifically the cloud-fog environment setup used for workflow scheduling in this research, computational resources are of three types, namely cloud servers, fog servers and end devices as illustrated in Fig. 2.4. Each of the resources consist of computing and storage capabilities along with memory, bandwidth and power requirements. The resources in the cloud and fog are represented as virtual machines (VMs). The end device is also included because for some small tasks, transferring them to the fog and cloud servers does not make sense from an

Table 2.1: Comparison between cloud and fog model

Requirement	**Cloud model**	**Fog model**
Latency	High	Low
Location of servers	Within the Internet	At the edge of the local network
Distance between client and server	Multiple hops	One hop
Attack on data	High probability	Low probability
Location awareness	No	Yes
Deployment	Centralized	Distributed

economic and resource utilization perspective.

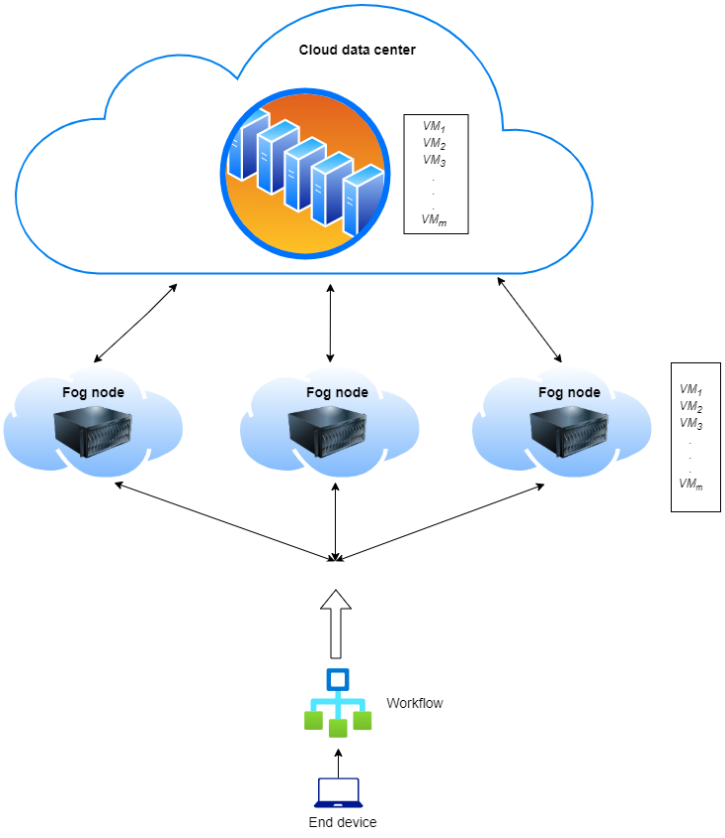


Figure 2.4: Cloud-Fog paradigm for workflow scheduling

2.4 Population-based Algorithms

Population-based algorithms are a category of metaheuristic algorithms that involve multiple candidate solutions evolving concurrently [20, 32]. These algorithms are known to produce better results than deterministic algorithms with regards to quality, and find approximate solutions faster than traditional exhaustive algorithms in terms of the computation time. Population-based techniques often use population characteristics to guide the search process which is a collective behavior of decentralized, self-organizing agents in a population or swarm [33].

The following sub-sections review the different population-based optimization algorithms that are evaluated in this work for scheduling scientific workflows.

2.4.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm introduced by Kennedy and Eberhart [33]. The technique is used to solve optimization problems from drawing inspiration by the social behavior of bird flocking, fish schooling and other animal societies that cooperate and share information to improve their position without relying on a leader. Over the years, the PSO algorithm has been successfully applied in a variety of fields, such as constrained mixed-variable optimization problems [34] and wireless sensor networks [35].

This technique uses a population of N individuals, represented as particles in the H-dimensional solution space. The current position of the i-th particle is represented as $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,H})$ and it's velocity is represented as $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,H})$. The quality of each particle is measured using a defined fitness function depending on the optimization problem. Each particle's movement is based on its best known personal position $\mathbf{pBest}_i$, and also moves towards the best known global position **gBest** for the entire swarm. This process leads the swarm to the best position over a number of iterations in the search process. At the k-th iteration, the elements of the particle's velocity and position are updated by using

$$\mathbf{v}_{i,h} = \omega \mathbf{v}_{i,h} + c_1 r_1 (\text{pBest}_{i,h} - \mathbf{x}_{i,h}) + c_2 r_2 (\text{gBest}_h - \mathbf{x}_{i,h}) \tag{2.1}$$

$$\mathbf{x}_{i,h} = \mathbf{x}_{i,h} + \mathbf{v}_{i,h}, \tag{2.2}$$

where $h = 1, 2, \ldots, H$; ω is the inertia weight; r_1 and r_2 are random numbers between (0,1); and c_1 and c_2 are the learning factors.

Algorithm 1: PSO Algorithm

```
1  Input: N, c1, c2, ω, and G;
2  Output: gBest and F(gBest);
3  Randomly generate N particles;
4  F(gBest) ← 0;
5  for i ← 1, N do
6      Compute the fitness function value, F(x_i), for particle i;
7      pBest_i ← x_i;
8      F(pBest_i) ← F(x_i);
9      if F(x_i) > F(gBest) then
10         gBest ← x_i;
11         F(gBest) ← F(x_i);
12 k ← 0;
13 while k ≤ G do
14     for i ← 1, N do
15         Update v_i and x_i by using Eq. 2.1 and Eq. 2.2;
16         Compute the fitness function value, F(x_i);
17         if F(x_i) > F(pBest_i) then
18             pBest_i ← x_i;
19             F(pBest_i) ← F(x_i);
20         if F(x_i) > F(gBest) then
21             gBest ← x_i;
22             F(gBest) ← F(x_i);
23     k ← k + 1;
```

Algorithm 1 illustrates the PSO optimization process. Parameters, N and G, denote the number of particles and the number of generations respectively, while the other parameters have already been defined above. The algorithm starts with the initialization of N, c_1, c_2, ω, and G. It then proceeds by creating N particles, each of which is evaluated. The fitness function value is evaluated, and the

personal and global best values are determined. Thereafter, the algorithm goes into an iterative process for G generations. It updates the global best and the personal best values whenever it gets better values.

The PSO algorithm has received much attention from many researchers due to its relative simplicity, short run time and fast convergence [12], however PSO is also known to suffer from premature convergence, especially in high-dimensional search spaces.

2.4.2 Genetic Algorithm

Genetic algorithms (GA) [36] are a class of population-based algorithms that start with a population of randomly generated individuals, represented as chromosomes, and are advanced over a number of iterations toward better solutions by applying genetic operators such as selection, crossover and mutation analogous to the genetic processes occurring in nature. The standard genetic algorithm first encodes the parameters to generate a certain number of individuals to create the initial population. The algorithm uses the fitness function as the criterion to evaluate the performance of each individual. Genetic operators are used in the creation of new generation of individuals. After creating a new generation of the population at each iteration, the algorithm performs a fitness evaluation of the new individuals. The elitism procedure is applied in order to generate the new population by merging the initial population and children. After creating a new generation of the population, the algorithm keeps on performing genetic operations in order to generate new offsprings; the fitness functions for each of the offsprings are evaluated and the best individuals are maintained. This process continues until the end condition is met upon which the individual with the best fitness is returned. GA has been applied to a broad range of optimization problems in various domains such as supply chain management [37], data mining [38], astrophysics [39], polymer science and engineering [40, 41], among others.

Algorithm 2 illustrates an overview of the GA optimization process. The parameters, N and G, denote the number of chromosomes and the number of generations respectively. The algorithm starts with the initialization of N, G, crossover rate and mutation rate. On line 3, N chromosomes are generated, each of which are evaluated using a fitness function. Then, from line 5, the algorithm goes into an iterative process for G generations, passing the chromosomes between the following three genetic operators.

Selection: Chromosomes are selected to be included in the next generation of the population for the following iteration. Some chromosomes are excluded in the evolutionary process due to fitness or randomness of the selection mechanism.

Crossover: It is used to generate new chromosomes by changing the position of the genes for two selected chromosomes. The crossover operator is performed by selecting a random number in the range of the chromosome genes. This number represents a division point of each chromosome, dividing it into two parts. Finally, the crossover produces an offspring chromosome combining two parts of both chromosomes genes.

Mutation: It is used to modify the new chromosomes that are produced from the crossover operator with better fitness than the existing chromosomes. The mutation operator modifies the chosen chromosome from the selection operator, and the chance of the mutation is based on the mutation rate parameter. The mutation method begins with a number that is randomly generated to be less than or equal to the mutation rate.

Algorithm 2: GA Algorithm

```
1  Input: N, G, crossover rate, mutation rate;
2  Output: The fittest solution found;
3  Randomly generate N chromosomes;
4  Perform fitness evaluation;
5  while stopping criteria not met do
6      Selection;
7      Crossover;
8      Mutation;
9      Fitness evaluation of new population;
10     if elitism then
11         population[0] = fittest;
12     Next generation
```

GA has better exploratory capabilities than PSO; it, however, suffers from the problem of slow convergence. Hence, latest research has been focusing on hybridizing more than one standard algorithm to improve performance. [14, 42, 43].

2.4.3 Differential Evolution

Differential Evolution (DE) is a population-based algorithm proposed by Storn and Price [44]. It uses difference vectors between individuals in order to efficiently explore the search space. It begins by randomly generating the initial population, where each individual is a candidate solution represented as a vector of real numbers, within the boundaries $[\text{xmin}_j, \text{xmax}_j] : \mathbf{x}_{i,j}, i = 1 \cdots \text{NP}, j = 1 \cdots H$, where H is the problem dimension and NP is the population size.

The mutation, crossover and selection operations are applied during the evolution process of DE. The mutation operation generates a mutant vector $\mathbf{v}_i$ for each target vector $\mathbf{x}_i$ by using

$$\mathbf{v}_{i,j} = \mathbf{x}_{r1,j} + F(\mathbf{x}_{r2,j} - \mathbf{x}_{r3,j}), \tag{2.3}$$

where $r1$, $r2$, and $r3$ are randomly chosen indices from $[1, \text{NP}]$, F is the scaling factor in the range $[0, 1]$ and $r1 \neq r2 \neq r3 \neq i$.

The crossover operation combines the mutant vector $\mathbf{v}_i$ with the target vector $\mathbf{x}_i$ to produce a trial vector u_i by using the crossover probability $\text{CR} \in [0, 1]$. Therefore, the elements of the trial vector that are randomly chosen components from the mutant vector are presented by

$$\mathbf{u}_{i,j} = \begin{cases} \mathbf{v}_{i,j} & \text{if rand}(0,1) \leq \text{CR } \textit{or } j = j_{\text{rand}} \\ \mathbf{x}_{i,j}, & \textit{otherwise}, \end{cases} \tag{2.4}$$

where j_{rand} is a randomly chosen index from $[1, H]$. In the selection operation, each trial individual produced as a result is compared with the corresponding individual in the current population. If the new individual's fitness is better than the current one, the new individual will replace the current one in the next generation, otherwise the old one will remain. The process is continuously repeated where the strong individuals are retained, and the inferior individuals are eliminated, which guides the search to a near optimal solution.

Algorithm 3 illustrates the pseudo-code for the DE algorithm. Parameter G denotes the index of the current generation, while the other parameters have already been defined above. The algorithm starts with the initialization of NP, CR, F and G. It then proceeds to generate the first generation of solutions, represented by population P. After that the algorithm goes into an iterative process until a stopping criterion is satisfied. In the mutation step on line 9, the mutated vector v is produced from the vectors of selected individuals. In the following crossover step, a trial vector u is

created by selection of elements, either from the mutated vector v or the original vector x. Finally, the fitness $f(u)$ of a trial vector is evaluated by an objective function and compared to the fitness $f(x)$ of the corresponding vector with the stronger one selected for the next generation.

Algorithm 3: DE Algorithm

Input: NP, CR, F, and termination criteria;
Output: $\mathbf{x}_{\text{best}}$ as the best found solution;
$G = 0$, $\mathbf{x}_{\text{best}} = \{\}$;
Randomly generate population $P = (\mathbf{x}_{1,G}, ..., \mathbf{x}_{NP,G})$;
$P_{\text{new}} = \{\}$, $\mathbf{x}_{\text{best}}$ = best from population P;
while *stopping criteria not met* **do**
  for $i \leftarrow 1$, NP **do**
    $\mathbf{x}_{i,G} = P[i]$;
    $\mathbf{v}_{i,G}$ by mutation (2.3);
    $\mathbf{u}_{i,G}$ by crossover (2.4);
    if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ **then**
      $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G}$;
      else
      $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$;
    $P_{\text{new}} \leftarrow \mathbf{x}_{i,G+1}$
  $P = P_{\text{new}}, P_{\text{new}} = \{\}$, $\mathbf{x}_{best}$ = best from population P;
  $G \leftarrow G + 1$

It can be seen from the DE algorithm that there are three control parameters, which have to be manually set namely, NP, CR and F. The setting of these parameters is important for the performance of DE [45, 46, 47]. However, the fine-tuning of the control parameter values is time-consuming, therefore, many latest DE variants use self-adaptation mechanisms to determine these parameters [48, 49, 50, 51].

2.5 Population-based approaches to Workflow Scheduling

Over the recent years, various workflow scheduling algorithms have been proposed based on different optimization techniques with single objective, bi-objective or multiple conflicting objectives [14, 15, 20, 52]. In this subsection, a review of the literature is presented on workflow scheduling strategies which are designed using population-based approaches to achieve better workflow execution performance. Most of these approaches are based on meta-heuristic approaches for workflow scheduling on the traditional cloud environment.

A novel Particle Swarm Optimization (PSO) algorithm for workflow scheduling, was proposed in [12]. The updates to the canonical PSO algorithm attempted to address the issue of premature convergence, but the study only considers makespan and cost as objectives for scheduling on the cloud. The work in [13] presented a multi-objective workflow scheduling algorithm in the cloud environment. The focus was on the cloud architecture but it also only considered makespan and cost as objectives, excluding other important objectives like energy minimization. A hybrid GA-PSO algorithm was proposed in [14]. In the first stage, the GA is applied to the entire generated population for the first half of the total iterations, to evolve towards the optimal solution. The PSO is applied to the population for the following half of the iterations, whereby the resulting chromosomes are passed to the PSO algorithm for the second half of the iterations. The workflow scheduling was done in a cloud environment and considered makespan, cost and load balancing as objectives in their optimization. On the other hand, the work in [19] presented a multi-objective approach for data intensive workflows scheduling in the fog paradigm, which considered the response time, the cost and reliability as objectives. However, the multi-objective algorithm was based only on NSGA-II and only considered the three aforementioned objectives.

An Objective Discrete PSO for workflow scheduling in cloud computing was proposed in [53]. It was based on a Dynamic Voltage and Frequency Scaling (DVFS) technique for energy minimization. This work only considered the following objectives: makespan, cost and energy. A multi-objective optimization based workflow scheduling strategy based on PSO is presented in [54]. The objectives were reducing execution time and cost of the workflow while meeting deadline and budget QoS constraints. The work in [42] proposed a hybrid meta-heuristic cloud-based algorithm to minimize makespan and balance load on the cloud VMs. The hybrid algorithm was a combination of GA and PSO, in order to take advantage of the exploratory capabilities of GA and the faster convergence rate

of the PSO. In, [55] another hybrid heuristic for scheduling workflows was developed. This technique first prioritized the tasks then GA was used for improving the mapping of tasks to VMs. The experiments were conducted for makespan, failure rate and load balancing of resources on a cloud paradigm. The work in [56] proposed a bi-objective hybrid workflow scheduling technique composed of the Gravitational Search Algorithm (GSA) with a Heterogeneous Earliest Finish Time (HEFT) heuristic. The hybridization of GSA and HEFT utilizes the exploitation abilities of both these algorithms in order to find better solutions. However, only two objectives were considered namely, makespan and total computational cost. Furthermore, the classic cloud computing paradigm was used in this work.

In [57], a Reliability-Driven (RD) reputation technique was used to measure the reliability of a VM. Then, the RD reputation was used in the proposed algorithm called Look-Ahead Genetic Algorithm (LAGA) to optimize the makespan and the reliability. However, crucial workflow scheduling objectives such as cost and energy consumption were not considered in this study. Another workflow scheduling algorithm, called Granularity Score Scheduling (GSS) was developed in [58]. The GSS was compared with two well-known algorithms known as HEFT [59] and Performance Effective Task Scheduling (PETS) [60]. The simulation results show that the GSS outperformed HEFT and PETS; however, only two objectives, makespan and VM utilization, were used. Other important objectives such as cost, energy and load balancing were not considered.

In [61], a Catfish-PSO scheduling algorithm was used to minimize both makespan and execution. The algorithm performed better than the canonical PSO, especially for larger scientific workflows; however, only two objectives were considered. Furthermore, the traditional two-tier cloud architecture was used. A PSO-based budget constrained workflow scheduling algorithm in a cloud environment was proposed in [62], with the objective of minimizing makespan while adhering to the constraint of a user's budget. The algorithm does, however, have a high run time in order to produce solutions for extra-large sized workflows. In [63], a bi-objective based heuristic based upon the HEFT algorithm was developed. The algorithm minimizes execution time and cost under deadline and budget constraints. The simulation results show that it performed much better compared to another HEFT based technique [64]. The algorithm did not consider energy consumption or the VM load distribution on the cloud.

In [65], an algorithm, based on the canonical PSO, was proposed with the aim of efficiently distributing tasks among cloud resources to reduce monetary cost and makespan. The run time of the

algorithm was, however, high compared to other optimization methods. Another multi-objective PSO based scheduling technique for multi-cloud environments was proposed in [66]. The objectives are makespan and cost with a reliability constraint. A coding strategy was employed to generate the workflow task schedules. Then, thresholds were used to ensure that the particles are within the boundary of the constraints. A dynamic workflow scheduling strategy was presented in [67], that was developed by combining artificial neural networks with the NSGA-II algorithm. The study aimed to minimize makespan, cost, energy and imbalance on the cloud. In [68], another dynamic workflow scheduling technique was presented, which found the partial critical paths of a workflow using the Intelligent Water Drops (IWD) algorithm. The IWD used the critical paths found to allocate workflow tasks to suitable VM instances on the cloud, in order to minimize makespan and resource utilization while meeting the deadline constraint. An online workflow scheduling policy for the cloud architecture was developed in [69]. The objectives considered were reducing makespan, cost and assignment of workloads to unreliable VMs. Also, monetary and budget constraints were considered in this study. In [70], an energy-aware real-time workflow scheduling heuristic was proposed. The objectives were the minimization of energy and cost of the workflow. The heuristic was run over different real-time workflows on the cloud environment with various QoS requirements.

Based on the review of the state-of-art workflow scheduling techniques, it can be seen that most of these strategies focus on bi-objective optimization by minimizing only makespan and cost of the workflow, without considering other important objectives such as energy consumption and load balancing. Furthermore, most of these works do not consider the fog processing layer which has desirable characteristics for the processing of large workloads like scientific workflows. Furthermore, the canonical PSO, which is usually characterised by premature convergence is the mostly used approach for scientific workflow scheduling; other emerging population-based techniques such as Differential Evolution have not been explored in this domain. This research has been motivated by all of these observations.

2.6 Simulation Frameworks

Simulation tools have been widely used by researchers to model network infrastructures. In particular, there are currently many simulation toolkits for cloud and fog environments [71, 72, 73]. The CloudSim toolkit [74] is one of the most used toolkits to model and simulate the traditional

cloud computing environment. However, CloudSim can only simulate the execution of simple independent tasks and hence, it does not cater for the execution of interdependent tasks as required by scientific workflows. Therefore, WorkflowSim [71] was proposed, as an extension of CloudSim with the capability of executing workflows on the cloud. It also provides support for evaluating different computational offloading strategies and workflows of different structures, however WorkflowSim does not support simulation of a cloud-fog model.

As fog computing is a recently proposed computing paradigm, the available simulation toolkits are not very rich in functionality and each of the toolkits target specific metrics and objectives [72, 73]. The iFogSim toolkit as proposed in [75] is one of the most popular tools that can be used to model a fog network topology and the associated fog resources in different layers. Similar to CloudSim, iFogSim provides the capability to simulate the execution of independent tasks. It also supports computational offloading strategies which determine the specific layer that the tasks will be offloaded for processing. However, iFogSim does not support the execution of workflow tasks that are characterized by complex dependencies.

Since this work focuses on workflow scheduling on a three tier cloud-fog environment, the aforementioned simulators could not be used without extensive code changes to the toolkit. Therefore, the recently proposed FogWorkFlowSim [28] is used in this work. FogWorkflowSim combines the functionality of iFogSim [75] and WorkflowSim [71], providing a cloud-fog computing environment and workflow system. Furthermore, the FogWorkFlowSim [28] is chosen for this work as it provides a suitable simulation framework to implement and evaluate the performance of canonical and novel techniques for workflow scheduling. This simulator also does not require extensive setup or updating low-level configuration which enables experiments to be easily repeatable and conducted in a controlled manner.

2.7 Chapter Summary

This chapter started with presenting the concept of workflows and, in particular, scientific workflows from different application domains. Then, the cloud-fog computing paradigm is described along with the differences between the cloud and fog processing tiers. An overview of three well-known population-based algorithms used for the performance evaluations in this work is given namely, PSO, GA and DE. Furthermore, many population-based approaches applied to workflow scheduling

specifically, is reviewed from literature. It is observed that the majority of the techniques focused on optimizing the scheduling of workflows on the traditional cloud architecture, with two to three objectives at most, excluding crucial objectives. Finally, a summary of the evolution of simulation frameworks to model cloud and fog environments is presented. The next chapter presents the comparison of population-based algorithms for workflow scheduling in cloud-fog environments. The various simulations and discussion of the results are also presented.

Chapter 3

Population-based Algorithms for Workflow Scheduling in Cloud-Fog Environments

3.1 Introduction

This chapter presents a comparative evaluation of four population-based optimization algorithms for workflow scheduling in cloud-fog environments. These algorithms are as follows: PSO, GA, DE and GA-PSO. The GA-PSO from [14] is implemented and DE is newly introduced to workflow scheduling. FogWorkflowSim [28] is used as the simulation environment with the makespan, cost and energy consumption serving as performance metrics. This chapter is organized as follows: Section 3.2 presents the mathematical formulations for makespan, cost and energy consumption along with the weighted sum based objective function. Section 3.3 discusses the workflow scheduling optimization process. Section 3.4 presents the environment setup of the different simulations conducted, results and discussion. Finally, Section 3.5 concludes the chapter.

3.2 Formulation of the objective function

In the cloud-fog environment setup presented here, there are m computational resources which are of three types, namely cloud servers, fog servers and end devices. Each of the resources consists of computing and storage capabilities along with memory, bandwidth and power requirements. The resources in the cloud and fog are represented as virtual machines (VMs). The end device is included as well because for some small tasks, transferring them to the fog and cloud servers does not make sense from an economic and resource utilization perspective. The selection of the optimal resource is determined by three objectives as outlined below.

3.2.1 Makespan

The makespan of a workflow can be defined as the total execution time for the completion of an entire workflow. Thus, makespan MS is calculated as follows:

$$\mathrm{MS} = \max\{\mathrm{FT}_{t_i}, t_i \in T\} - \min\{\mathrm{ST}_{t_i}, t_i \in T\} \tag{3.1}$$

where ST_{t_i} and FT_{t_i} are the starting time and finishing times respectively for task t_i in a particular workflow.

3.2.2 Cost

This metric consists of computation cost and communication cost. Computation costs apply for all the three computational resource types while communication costs are not included when the tasks are executed on the end device. The computational cost [12] when using computing resource r is defined as

$$\mathrm{CE}_i^r = \mathrm{pr} \cdot (\mathrm{FT}_{t_i} - \mathrm{ST}_{t_i}), \tag{3.2}$$

where pr is the unit processing cost. The communication cost refers to the data transfer cost of a task output of size d_{ij} from the resource processing task i to the resource allocated to process task j is determined by

$$\mathrm{CC}_{ij} = \mathrm{trc}_{ij} \cdot d_{ij}, \tag{3.3}$$

where trc_{ij} is the unit cost of communication from the resource, where task i is mapped, to the resource, where task j is mapped; $\text{trc}_{ij} = 0$ when the two tasks are executed on the same resource. Therefore, the total cost TC is

$$\text{TC} = \sum_{i=1}^{n}\sum_{j=1}^{n}\text{CC}_{ij} + \sum_{r=1}^{m}\sum_{i=1}^{n}\text{CE}_i^r. \tag{3.4}$$

3.2.3 Energy Consumption

The energy consumption model [53] is constructed using active E_{active} and idle E_{idle} components. The former refers to the energy consumed when a task is being executed while the latter is the energy used when a resource is idling. The active energy is calculated by

$$E_{\text{active}} = \sum_{i=1}^{n}\alpha f_i v_i^2(\text{FT}_{t_i} - \text{ST}_{t_i}), \tag{3.5}$$

where α is a constant; f_i and v_i are the frequency and supply voltage for the resource on which task i is being executed. During the idle, the resource goes into sleep mode, where the voltage supply level and the relative frequency are at the lowest level. Therefore, the energy consumed during the idle period is determined by using [53]:

$$E_{\text{idle}} = \sum_{j=1}^{m}\sum_{\text{idle}_{jk}\in\text{IDLE}_{jk}}\alpha f_{min\ i} v_{min\ i}^2 L_{jk}, \tag{3.6}$$

where IDLE_{jk} is a set of idling slots on resource j, $f_{min\ i}$ and $v_{min\ i}$ refer to the frequency and lowest supply voltage on resource j respectively; L_{jk} is the duration of idling time for idle_{jk}. The total energy TE consumed on the cloud-fog system for the execution of the entire workflow is

$$\text{TE} = E_{\text{active}} + E_{\text{idle}}. \tag{3.7}$$

Therefore, weighted sum objective function, that incorporates makespan, cost and energy consumption, is defined by:

$$F(\mathbf{p}) = w_1 \cdot \mathrm{MS}_{\mathrm{norm}} + w_2 \cdot \mathrm{TC}_{\mathrm{norm}} + w_3 \cdot \mathrm{TE}_{\mathrm{norm}}, \tag{3.8}$$

where $\mathbf{p}$ is the assignment of the n tasks of a workflow to the m available computing resources. $\mathrm{MS}_{\mathrm{norm}}$, $\mathrm{TC}_{\mathrm{norm}}$ and $\mathrm{TE}_{\mathrm{norm}}$ are the normalized makespan, total cost and total energy respectively; w_1, w_2 and w_3 are the respective weights that determine the contribution of each of them to the overall objective function. Equal weighted coefficients are used here to obtain a balanced contribution of the three objectives since all are equally important in a good solution. Normalization is used here in order to eliminate any biases in the realized objective function.

3.3 The Workflow Scheduling Optimization Process

This section begins by describing how the workflow tasks are mapped to the available resources and how this mapping is used to generate a solution vector. The second part describes the optimization process based on GA, PSO, DE and GA-PSO.

3.3.1 Mapping of workflow tasks to computational resources and generation of the solution vector

The workflows in this work can be scheduled for execution at the source end device, at the fog server or at the cloud server. Each of these computation resources has its own computational power and access bandwidth with respect to the end node. The end nodes do not offload their tasks to fellow end nodes; they can only offload their tasks to fog and cloud servers. Therefore, in the scheduling of workflows, only one representative end device for each of the end nodes is incorporated in the encoding process.

Workflow scheduling in cloud-fog computing environments is a discrete problem, hence natural numbers are used to encode the individuals for each populated-based algorithm. In the case of the GA, PSO and DE, the individuals represented by the chromosome, particle and agent respectively, are mapped to possible task-resource schedules. The dimension or length of each individual is n, which is the total number of tasks in the workflow; each position in the individual's vector is a

positive integer representing the task number. The value assigned to this position is the virtual machine ID that is allocated to execute the task. The ID numbers are selected from the virtual machines available on the three layers of the cloud-fog architecture. Suppose a workflow has 10 tasks which are scheduled for execution on six available virtual machines, specifically one end device, two fog nodes and three cloud servers. In this instance, the individual's length is 10 and each element is an integer between 1 and 5. An example task assignment of this individual can be expressed as $\mathbf{p} = \{4,5,1,6,2,3,4,5,1,6\}$. Fig. 3.1 shows a graphical illustration of this example. A tabular representation of the individual's schedule is illustrated in Table 3.1 and Table 3.2.

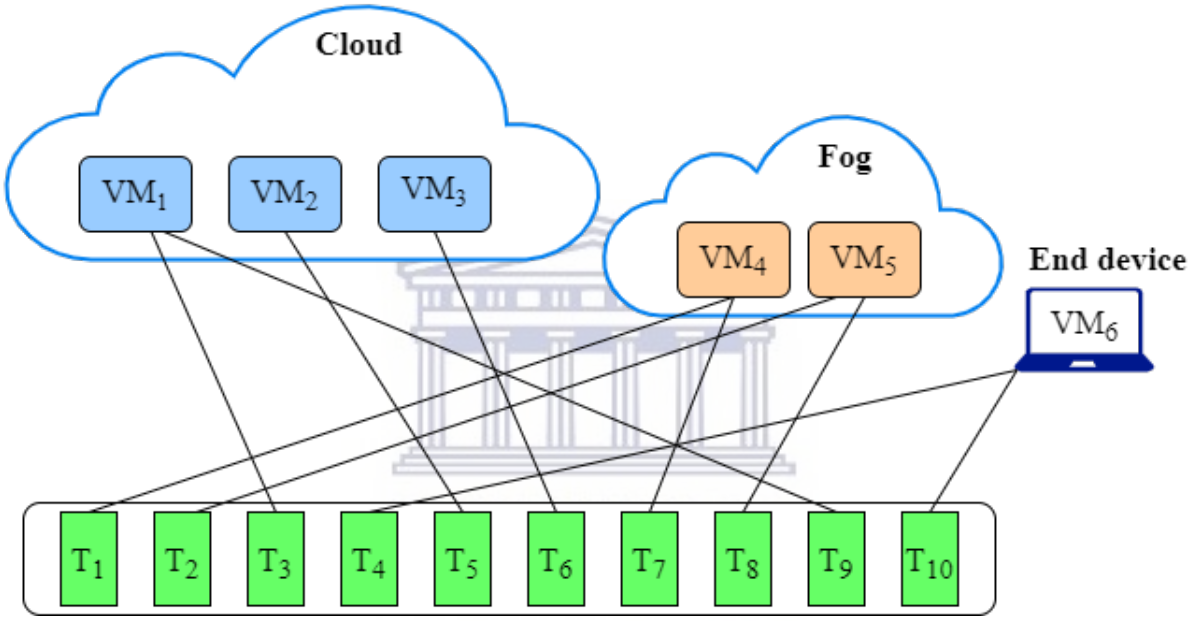


Figure 3.1: Example of a task-resource mapping on the cloud-fog environment

Table 3.1: Example of the Individual's encoded schedule

Task number	1	2	3	4	5	6	7	8	9	10
VM ID	4	5	1	6	2	3	4	5	1	6

Table 3.2: Example of the Task-Resource Allocation on the Cloud-Fog Layers

VM Layer	Cloud	Cloud	Cloud	Fog	Fog	End
VM ID	1	2	3	4	5	6
Assigned Task	3,9	5	6	1,7	2,8	4,10

3.3.2 The Optimization Process

The implementation of the four optimization algorithms are described in this subsection. The candidate solutions are generally called vectors for all algorithms. Specifically, in PSO parlance, these vectors are called particles and in GA, they are called chromosomes. Fig. 3.2 shows the generic flowchart for a population-based scheduling algorithm.

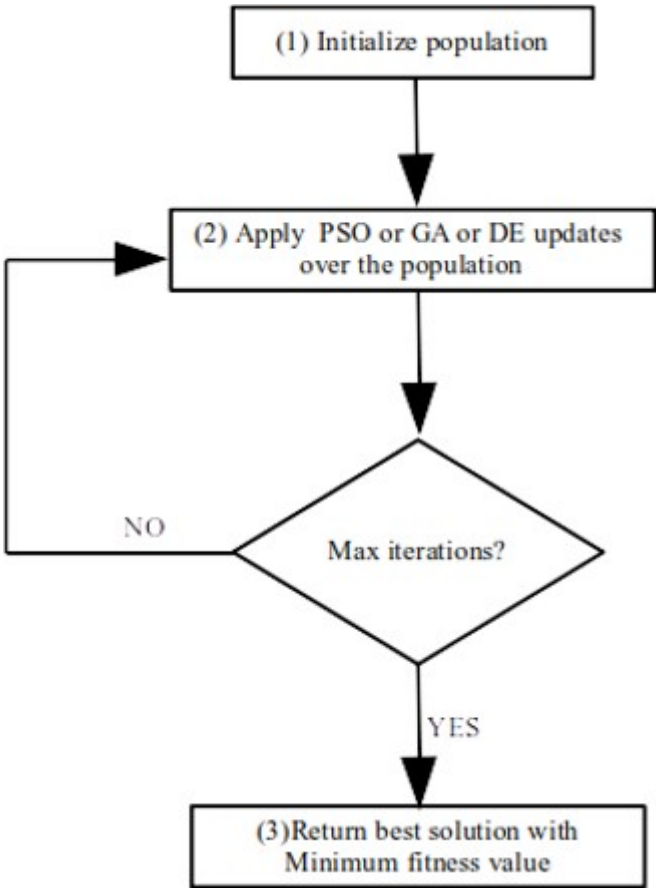


Figure 3.2: Flowchart of a generic population-based algorithm for workflow scheduling

In the initialization step, a population of P individuals of length n is initialized with random integer values depicting the possible VM unto which the corresponding task would be assigned. Then, the workflow simulation for each individual is carried out; this is followed by the determination of the three performance metrics, namely: makespan MS, cost TC and energy TE as described in Section 3.2. The minimum and maximum values values of the metrics (MS_{min}, MS_{max}, TC_{min}, TC_{max}, TE_{min} and TE_{max}) are determined for this initial population, and based on these values, normalized metrics are calculated for each vector by using

$$\text{MS}_{\text{norm}} = \frac{\text{MS}_i - \text{MS}_{\text{min}}}{\text{MS}_{\text{max}} - \text{MS}_{\text{min}}}$$

$$\text{TC}_{\text{norm}} = \frac{\text{TC}_i - \text{TC}_{\text{min}}}{\text{TC}_{\text{max}} - \text{TC}_{\text{min}}} \tag{3.9}$$

$$\text{TE}_{\text{norm}} = \frac{\text{TE}_i - \text{TE}_{\text{min}}}{\text{TE}_{\text{max}} - \text{TE}_{\text{min}}}$$

where MS_i, TC_i and TE_i are the makespan, cost and energy values for vector $i \in \{1, ..., P\}$. The fitness function F_i for the individual is determined by using the weighted sum objective function in (3.8). After this, the other post initialization processes are more specific for each algorithm. In the GA approach, the worst and the best solutions for the population are saved. In the PSO approach, $\forall i \in \{1, ...P\}$, $\text{pBest}_i = F_i$ and $\text{gBest}_i = \min\{\text{pBest}_i\}$. In the DE approach, the best solution for the population is saved.

In the second step, algorithm specific routines are applied to the population as follows:

1. In the GA approach, selection, crossover and mutation operators are applied in order to generate a new pool of potential solutions that are evaluated on a workflow simulation. The fitness values for all chromosomes are determined. The elements of the best chromosome and its fitness value are saved. Elitism is applied to truncate the pool of potential solutions to only the best ones, in preparation for the iteration.

2. In the DE approach, mutation and crossover are performed. The fitness values for all chromosomes are determined. The elements of the best agent and its fitness value are saved. Some of the well performing offsprings replace some agents in the original solution, thereby creating room for further improvements in the next iteration.

3. In the PSO approach, the particle's positions are updated using the PSO equations in (2.1) and (2.2). The fitness values for all particles are determined. The elements of the best particle and its fitness value are saved. If the new particle positions give a better fitness value than its pbest, the pbest is updated to the particle's new position; the fitness value is updated accordingly. If one of the particles' updated pbest is better than the gbest, the gbest is updated accordingly, in preparation for the next iteration.

4. The GA-PSO approach applies GA's update mechanism for the first half of the iteration. The PSO update process is applied until the end. If the maximum number of iterations has not been reached, the algorithm goes back to the second step. This process keeps on getting repeated for each of these algorithms until the maximum number of iterations is reached. Then the best solution with the lowest fitness value is saved in step 3.

The GA and PSO codes are already incorporated into the FogWorkflowSim Simulator [28]. The DE and GA-PSO algorithms, as well as the weighted sum based objective function proposed have been integrated for this work.

3.4 Simulations

Two sets of simulation experiments are conducted in this section. The first set, which is presented in subsection 3.4.1, focuses on a comparative evaluation of GA, PSO, GA-PSO and DE for the problem of scientific workflow scheduling. The second set, which focuses on the comparative evaluation of performance of the PSO algorithm under cloud and cloud-fog environments, is presented in section 3.4.2.

The simulations are conducted using the FogWorkflowSim simulator which is executed using the Eclipse Java IDE. The hardware used is a Dell-Inspiron laptop with the system specifications as shown below in Table 3.3.

Table 3.3: System Specifications

Specification	**Description**
Processor	Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz
Installed memory (RAM)	16GB
System type	64-bit Operating System, x64-based processor
Operating System	Windows 10 Professional

The following parameter settings of the experiments are applicable to both subsection 3.4.1 and 3.4.2: The population size is set to 50 for each algorithm. The PSO learning factors $C_1 = C_2 = 2$. The PSO inertia weight is 1. The number of iterations for all algorithms is 100. The weighted coefficients w_1, w_2 and w_3 are equal.

3.4.1 Comparative Evaluation of GA, PSO, GA-PSO and DE for Scientific Workflow Scheduling

This subsection presents a comparative evaluation of GA, PSO, GA-PSO and DE as workflow scheduling optimization tools. In the simulations conducted in this subsection, each of these techniques are used to produce a solution vector which is an optimized task-resource mapping for scheduling a scientific workflow according to eq. (3.8). The comparative evaluation of the performance objectives for the realized solution vectors will provide scientific practitioners from various industry and research domains insight into which algorithm may be more suited to their use case and hence, offers optimal scheduling. This will enable these workflows to be scheduled more efficiently and reduce economic cost associated with utilizing cloud and fog infrastructure.

The parameter settings of the experiments conducted here are as follows: The GA crossover and mutation rates are 0.8 and 0.1 respectively. The DE crossover probability is 0.4 and the differential weight is 1.2. The algorithms are evaluated using three scientific workflows namely Montage, Epigenomics and CyberShake, with three different task amounts per workflow. The simulations are performed 10 times for each workflow to get the average performance of the algorithms. Three end devices, 5 fog VMs and 5 cloud VMs are used. The characteristics for each server on the three cloud-fog layers along with the parameter settings for the simulation environment are shown in Table 3.4.

Table 3.4: Parameter Settings of Cloud-Fog Computing Environment

Parameters	**End device**	**Fog node**	**Cloud server**
Processing rate (MIPS)	1000	1300	1600
Task execution cost ($)	0	0.48	0.96
Communication cost ($)	0	0.01	0.02
Working power (mW)	700	800	1600
Idle power (mW)	30	400	1300
Uplink bandwidth (Mbps)	20	10	1
Downlink bandwidth (Mbps)	40	10	10

In Figs. 3.3-3.5 show the results for makespan, cost and energy consumption for the Montage workflow. As expected, all the metrics increase as the number of tasks increases. Results show that PSO is exhibiting poorer performance compared to the other three approaches. On the other hand,

all the three other approaches exhibit similar levels of performance. This is probably because of PSO's well-known problem of premature convergence and getting trapped in the local minimum. Clearly, the GA-PSO algorithm, which adopts the GA approach in the first half of the run and the PSO approach in the latter iterations, seems to benefit from the GA's ability to see a wide range of solutions, courtesy of the random mutation and crossover operators.

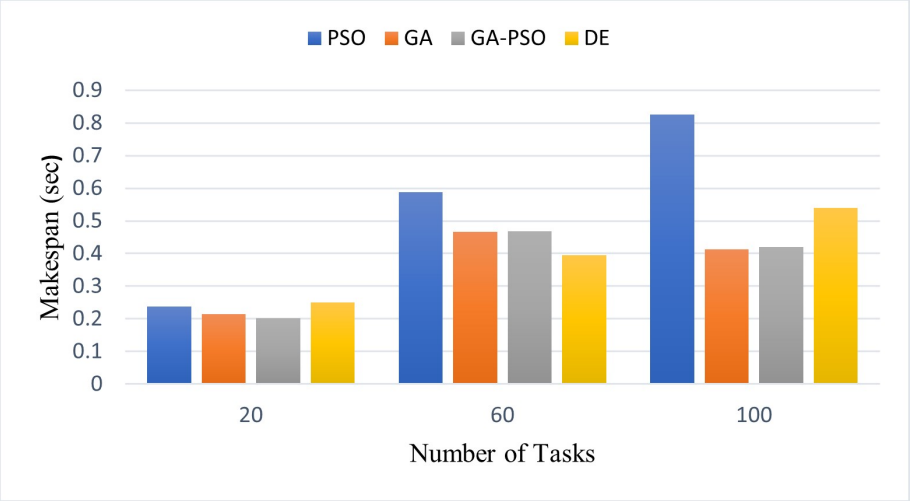


Figure 3.3: Makespan for Montage

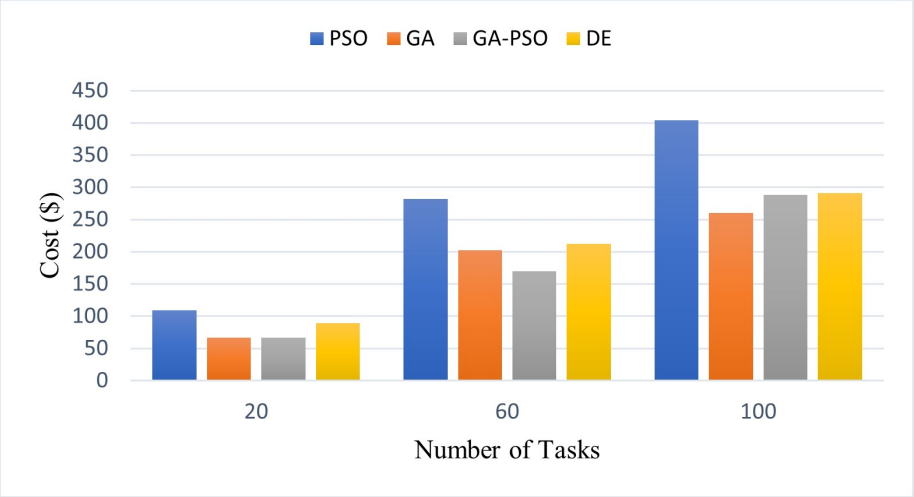


Figure 3.4: Total cost for Montage

In Figs. 3.6-3.8 show the results for makespan, cost and energy consumption for the Epigenomics workflow. The metrics for 24 and 47 tasks is very low but it springs up when the number of tasks increases to 100. For instance, makespan increases from less than 10 sec for 24 and 47 tasks to beyond 50 sec at 100 tasks; cost increases from less than $10 000 for 24 and 47 tasks to beyond

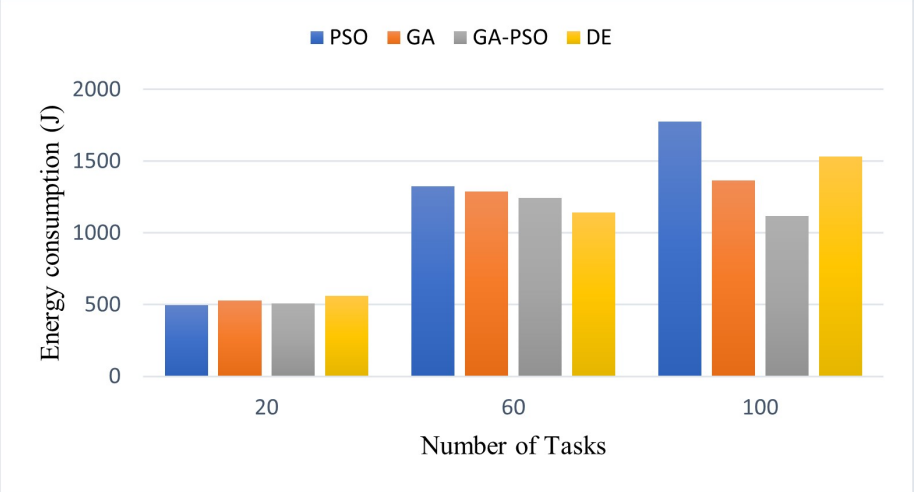


Figure 3.5: Energy consumption for Montage

$50 000 at 100 tasks; energy increases from less than 50 000J for 24 and 47 tasks to beyond 250 000J at 100 tasks. This is due to the map jobs in the Epigenomics workflow [1] responsible for aligning sequences with the reference genome, which become significantly more computationally intensive with higher runtimes as the number of tasks increase. In terms of performance, PSO is still exhibiting poorer performance compared to the other three approaches. There is, however, improvement in terms of energy consumption as it performs better than the other approaches for that metric.

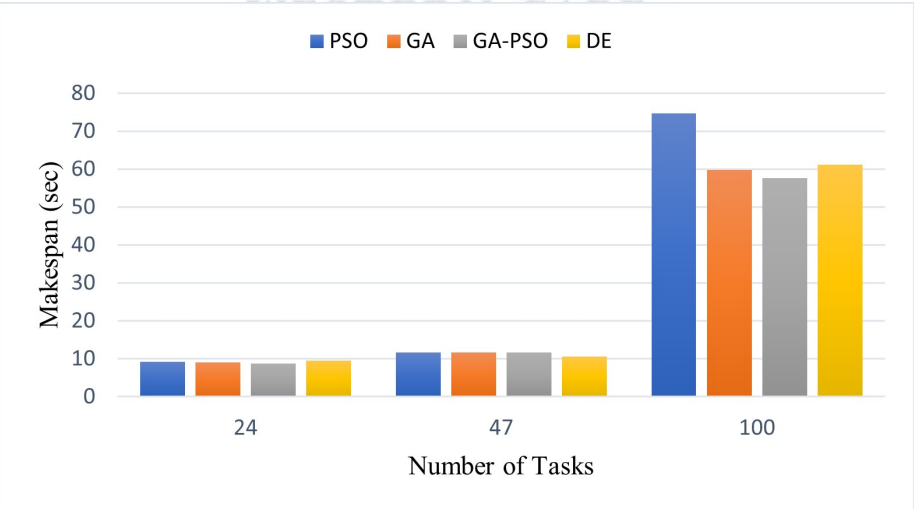


Figure 3.6: Makespan for Epigenomics

In Figs. 3.9-3.11 show the results for makespan, cost and energy consumption for the CyberShake

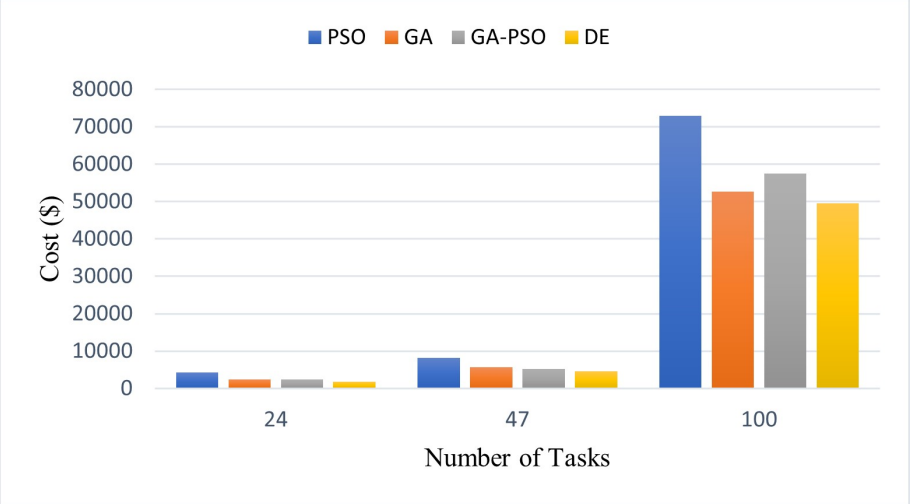


Figure 3.7: Total cost for Epigenomics

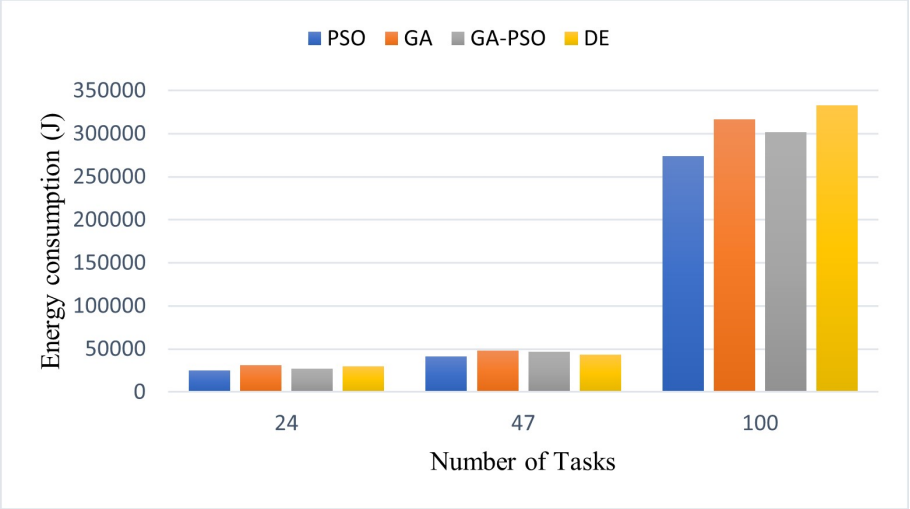


Figure 3.8: Energy consumption for Epigenomics

workflow. In Fig. 3.9, the makespan performance remains fairly constant for all algorithms. GA-PSO exhibits the best performance for all tasks while PSO seems to perform better than in the previous two workflows. DE's makespan increases as the number of tasks increases. In Fig. 3.10, GA-PSO generally continues to perform better than the other approaches for 30 and 50 tasks. DE's cost goes down drastically as the number tasks increases. It looks like the DE algorithm allocates more tasks to the end devices as the number of tasks increases thereby reducing the cost greatly. There are very low data transmission and computation costs due to less usage of cloud and fog servers. This leads to high energy consumption on the end devices as shown by Fig. 3.11.

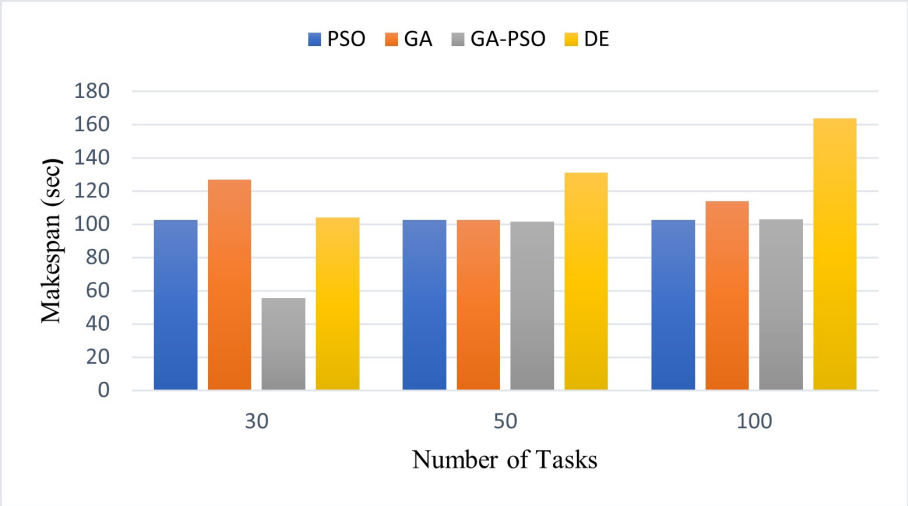


Figure 3.9: Makespan for CyberShake

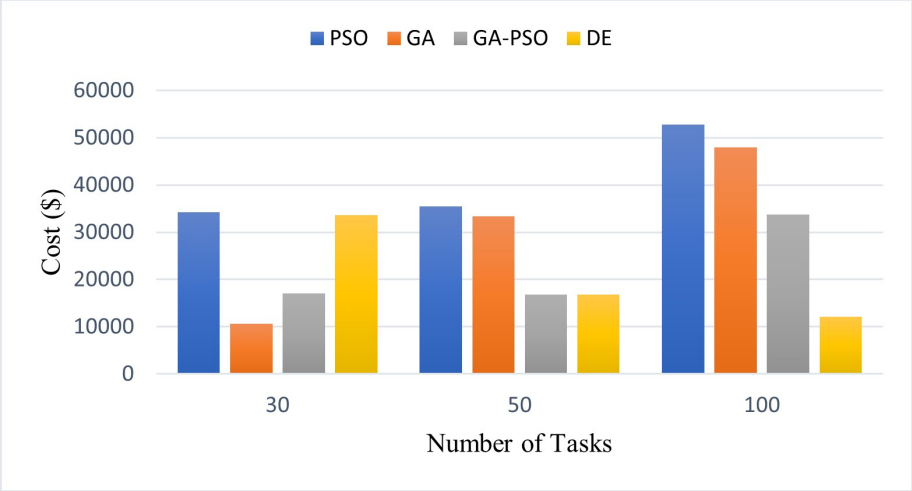


Figure 3.10: Total cost for CyberShake

The general outcome of these results is that there is no single algorithm that stands out among these algorithms even though the GA-PSO approach seems to exhibit slightly better performance. This shows that there is room for improvement if hybrid algorithms are proposed for the workflow scheduling problem. In terms of practical implementation, the solution vector realized from the optimization process can be downloaded into the end device as a lookup table for scheduling the workflows.

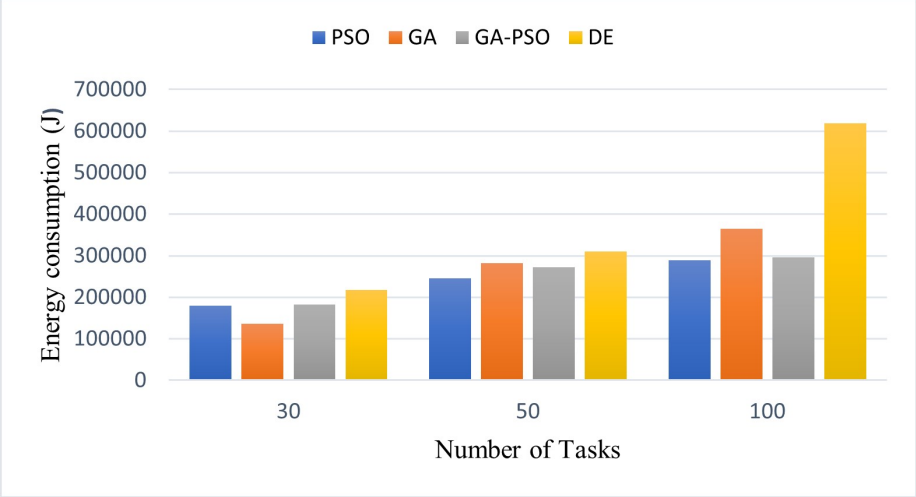


Figure 3.11: Energy consumption for CyberShake

3.4.2 Comparative Evaluation of PSO-based Scientific Workflow Scheduling in Cloud and Cloud-Fog Environments

This subsection presents the performance evaluation of the canonical PSO algorithm, applied to both the traditional cloud and the emerging three-tier cloud-fog environment. These experiments are carried out to compare the PSO workflow scheduling performance under the two environments, and to investigate the incorporation of the fog layer for the execution of workflows and its effect on processing efficiency and energy consumption.

Table 3.5: Parameter Settings Of Simulation Environment

Parameters	**Fog Server**	**Cloud Server**
Processing rate (MIPS)	1000	2000
Task execution cost ($)	0.48	0.96
Communication cost ($)	0.01	0.02
Working power (mW)	700	1700
Idle power (mW)	200	1200
Uplink bandwidth (Mbps)	500	300
Downlink bandwidth (Mbps)	800	500

The parameter settings of the experiments conducted here are as follows: The Montage, CyberShake and Epigenomics scientific workflows with 500 tasks each are used as input. The simulations are

performed 10 times for each workflow and environment setup. The number of cloud servers and fog servers are 10 and 6, respectively. The characteristics of each server on the two tiers along with the parameter settings for the simulation environment are shown in Table 3.5.

In Figs. 3.12-3.14, the results for makespan, cost and energy consumption for the Montage workflow for 500 tasks are illustrated. The makespan is lower for the cloud-fog environment, as expected, due to the 6 additional fog servers used in the simulation. The cost metric is significantly better in the cloud-fog layers. This is likely due to the reduced processing and data transfer costs associated with the fog layer. The energy consumption is also reduced in cloud-fog as the larger size of the cloud requires more energy to remain online, and utilizing the fog with the cloud enables more efficient and distributed processing of the workflow tasks.

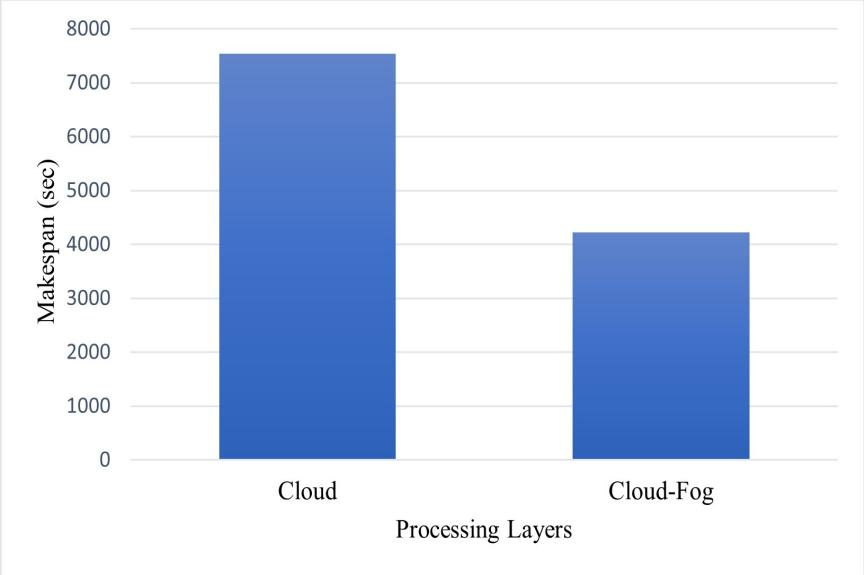


Figure 3.12: Makespan for Montage

In Figs. 3.15-3.17, the results for makespan, cost and energy consumption for the CyberShake workflow for 500 tasks are illustrated. The performance metric comparative results for the cloud and cloud-fog are similar to what was observed for the Montage workflow, however, the metric values are significantly higher. This is because the CyberShake workflow task sizes and task runtimes are much higher compared to Montage.

In Figs. 3.18-3.20, the results for makespan, cost and energy consumption for the Epigenomics workflow for 500 tasks are illustrated. The performance metric comparative results for the cloud and cloud-fog are similar to what was observed for the CyberShake workflow, with significantly high

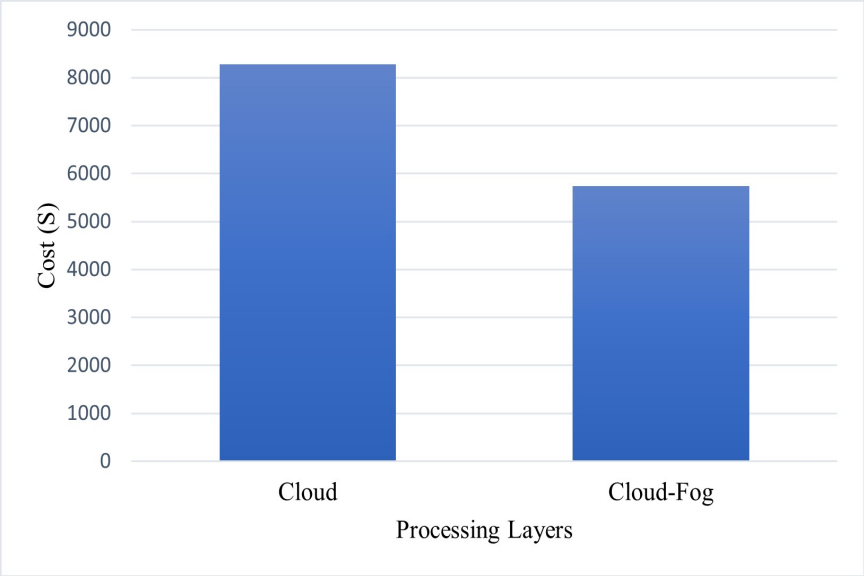


Figure 3.13: Total cost for Montage

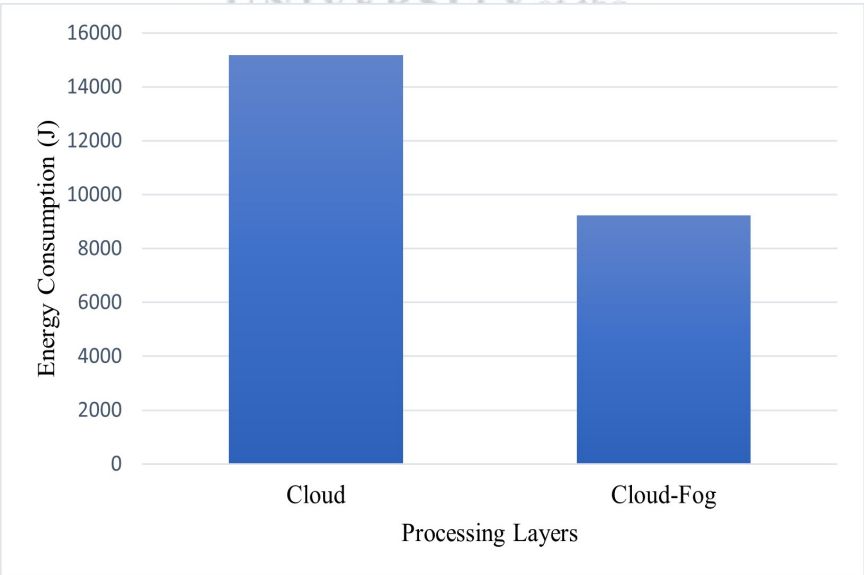


Figure 3.14: Energy consumption for Montage

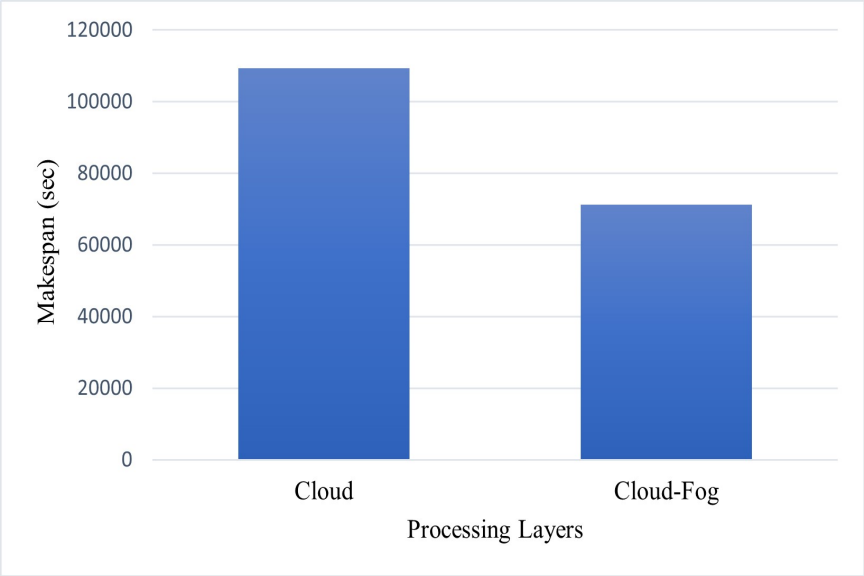


Figure 3.15: Makespan for CyberShake

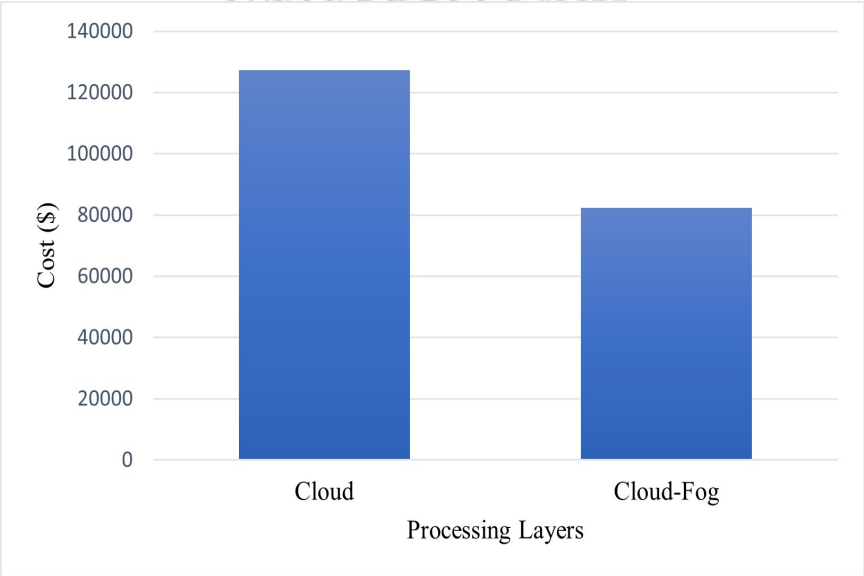


Figure 3.16: Total cost for CyberShake

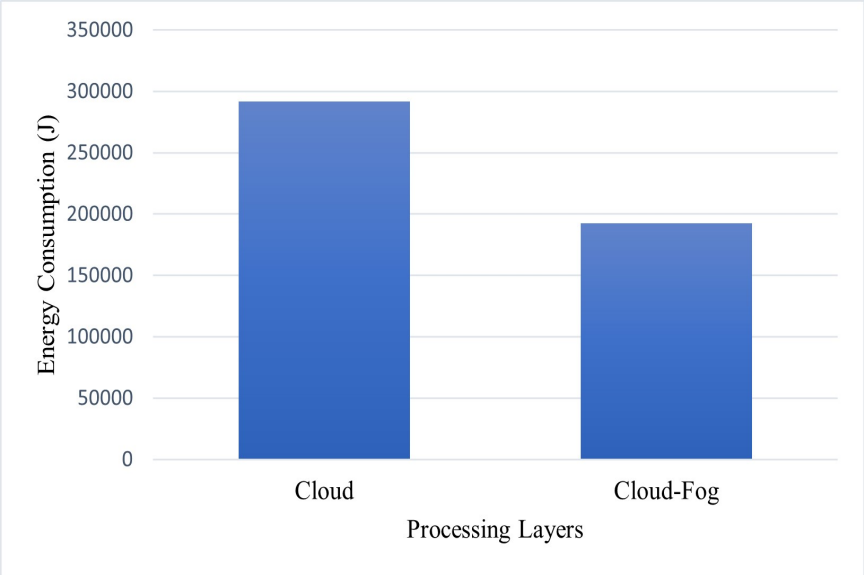


Figure 3.17: Energy consumption for CyberShake

metric values. Also, this is due to the Epigenomics workflow responsible for aligning sequences with the reference genome [1], which become significantly more computationally intensive with higher runtimes as the number of tasks increase. In terms of performance, PSO is still exhibiting better performance on the cloud-fog environment compared to the traditional cloud.

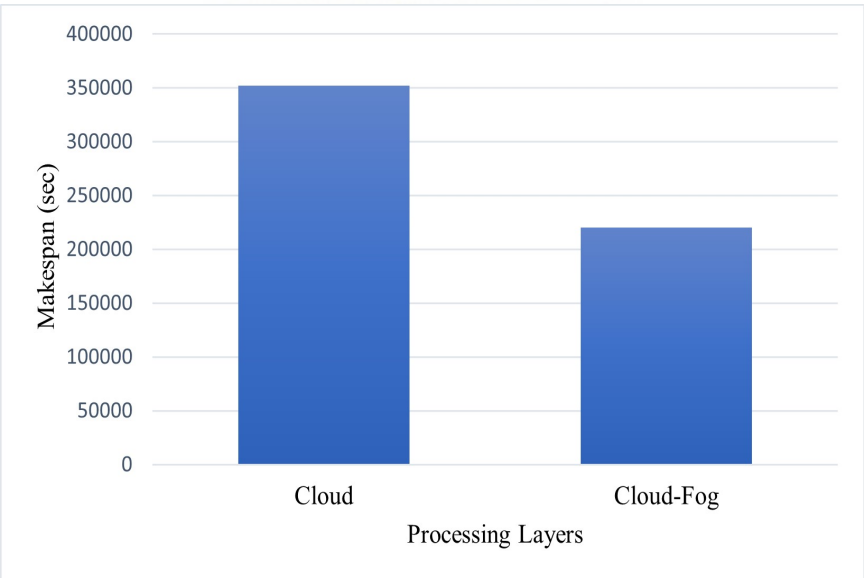


Figure 3.18: Makespan for Epigenomics

The results show that the cloud-fog environment performed better than the cloud especially in

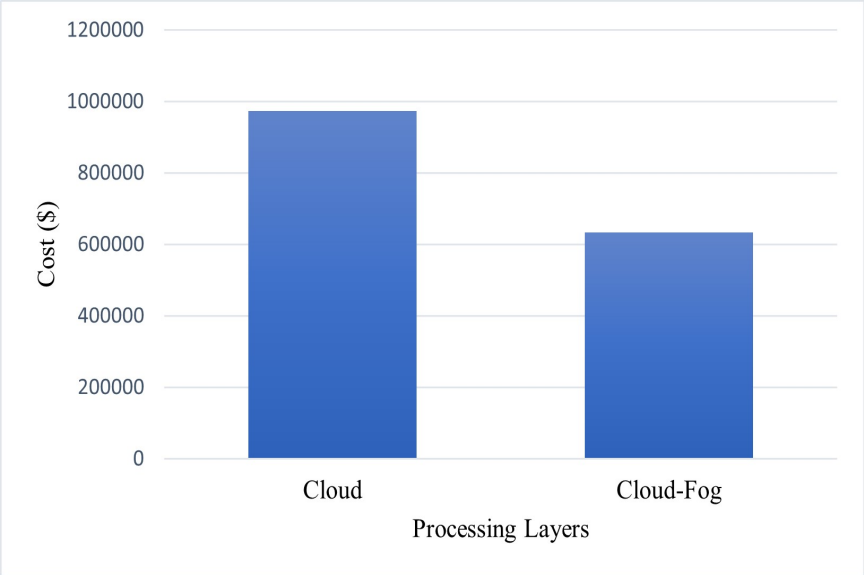


Figure 3.19: Total cost for Epigenomics

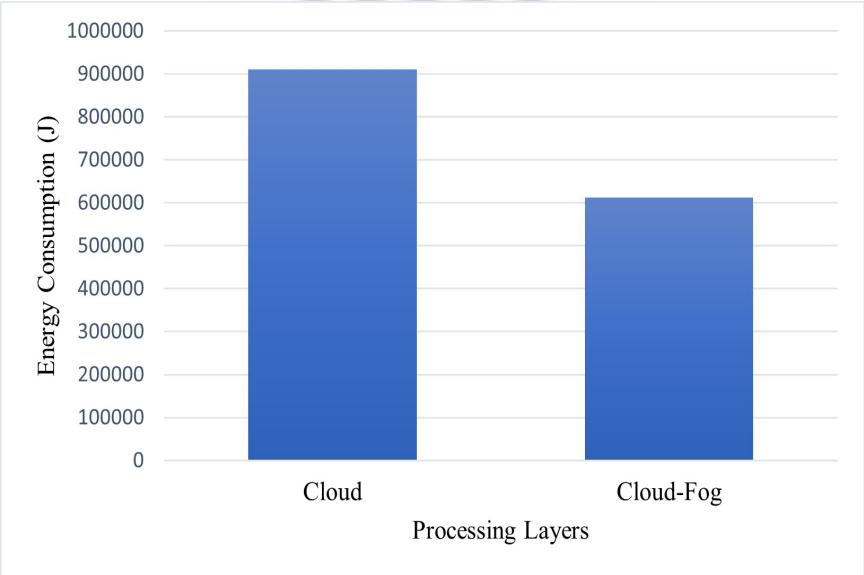


Figure 3.20: Energy consumption for Epigenomics

terms of overall cost and energy consumption. Therefore, the incorporation of the fog layer for the execution of workflows has the potential to improve processing efficiency justifying the benefits of the emerging cloud-fog computing paradigm. The adoption of the fog layer becomes more important for workflow scheduling as the demand for cloud services grow, since increases in latency and bandwidth requirements of the cloud will make the scheduling of large-scale workflows impractical.

3.5 Chapter Summary

In this chapter, the formulation of the objective function made up of makespan, cost and energy is presented along with the workflow scheduling optimization process. The mapping of workflow tasks to computational resources and generation of the solution vector is also discussed. The results show that the hybrid combination of the GA-PSO algorithm exhibits slightly better performance than the standard algorithms. In addition, the comparative evaluation of cloud and cloud-fog environments using PSO indicate that the incorporation of the fog layer for the execution of workflows has the potential to improve processing efficiency and reduce energy consumption, motivating the cloud-fog computing paradigm. The next chapter will present the proposed Multi-Swarm PSO for scientific workflow scheduling in cloud-fog environments.

Chapter 4

Multi-Swarm PSO for Workflow Scheduling in Cloud-Fog Environments

4.1 Introduction

This chapter presents a Multi-Swarm Particle Swarm Optimization (MS-PSO) algorithm to improve the scheduling of scientific workflows in cloud-fog environments. MS-PSO seeks to address the canonical PSO's problem of premature convergence, which leads it to suboptimal solutions. In MS-PSO, particles are divided into several swarms, with each swarm having its own cognitive and social learning coefficients. This work also expands the weighted sum objective presented in chapter 3 to include load balancing for cloud and fog tiers. The MS-PSO approach is compared with the canonical PSO, Genetic Algorithm (GA), Differential Evolution (DE) and GA-PSO. This chapter is organized as follows: section 4.2 presents the addition of load balancing to the objective function. Section 4.3 presents a brief review of the state-of-the-art MS-PSO algorithms and techniques. Section 4.4 describes the proposed Multi-objective Workflow Scheduling based on MS-PSO while Section 4.5 discusses the performance evaluation. Finally, Section 4.6 concludes the chapter.

4.2 The addition of load balancing to the objective function

Load balancing of the respective cloud and fog layers is added to the objective function presented in Chapter 3. Load balancing can be defined as the calculation of the standard deviation of the load of all the VMs. The aim is to ensure that all the VMs have almost equal loads. So, minimizing the standard deviation of the VMs produces improved load management among the VMs. The load of a VM is determined by the ratio between the length or size of the tasks executed by a VM and capacity of the VM. The fog layer VM characteristics are different compared to the cloud. Therefore to ensure fair overall load distribution, the standard deviation of the load is calculated independently for the cloud and fog layers. The end device is excluded when balancing the load on the cloud-fog environment.

Let c and f denote the number of VMs in the cloud and fog layers respectively. Hence, load balancing on the cloud is defined by

$$\mathrm{LBC} = \sqrt{\frac{\sum_{i=1}^{c}(\mathrm{LC}_i - \mathrm{ALC})^2}{c}} \tag{4.1}$$

where LC_i refers to the load of the i-th VM on the cloud, and ALC is the average load of all the VMs on the cloud layer. Similarly, the load balancing on the fog is defined by

$$\mathrm{LBF} = \sqrt{\frac{\sum_{i=1}^{f}(\mathrm{LF}_i - \mathrm{ALF})^2}{f}} \tag{4.2}$$

where LF_i refers to the load of the i-th VM on the fog, and ALF is the average load of all the VMs on the fog layer. Therefore, using the four aforementioned objectives, the weighted sum objective function is defined by:

$$\begin{aligned} F(\mathbf{p}) = {} & w_1 \cdot \mathrm{MS}_{\mathrm{norm}} + w_2 \cdot \mathrm{TC}_{\mathrm{norm}} + w_3 \cdot \mathrm{TE}_{\mathrm{norm}} \\ & + w_4 \cdot \mathrm{LBC}_{\mathrm{norm}} + w_5 \cdot \mathrm{LBF}_{\mathrm{norm}} \end{aligned} \tag{4.3}$$

where $\mathbf{p}$ is the assignment of the n tasks of a workflow to the m available computing resources presented in Chapter 3. The parameters $\mathrm{LBC}_{\mathrm{norm}}$ and $\mathrm{LBF}_{\mathrm{norm}}$ are the load balancing among cloud VMs, and load balancing among fog nodes respectively; w_* is the coefficient weight. Equal weights are used in the performance evaluations to obtain an equal contribution of each objective i.e. $w_* = 0.2$. Normalization removes any biases in the objective function and also ensures that there

is a balanced contribution from all the objectives. To obtain the normalized objective function parameters in equation (4.3), objective function values LBC_i and LBF_i are defined for the i^{th} potential solution. Similarly for makespan, cost and energy, the normalized objective for load balancing are defined by

$$\mathrm{LBC}_{\mathrm{norm}} = \frac{\mathrm{LBC}_i - \mathrm{LBC}_{\mathrm{min}}}{\mathrm{LBC}_{\mathrm{max}} - \mathrm{LBC}_{\mathrm{min}}}$$
$$\mathrm{LBF}_{\mathrm{norm}} = \frac{\mathrm{LBF}_i - \mathrm{LBF}_{\mathrm{min}}}{\mathrm{LBF}_{\mathrm{max}} - \mathrm{LBF}_{\mathrm{min}}} \tag{4.4}$$

where the max and min values represent the maximum and minimum of the objectives; these are determined from the initial population of the algorithms.

Now that this section has defined the objective function (Eq. (4.3)), the next section presents the multi-objective workflow scheduling based on the proposed multi-swarm PSO optimization algorithm.

4.3 A brief review of the state-of-the-art MS-PSO algorithms and techniques

In recent years, there have been many efforts to break down the PSO algorithm into sub-swarms, and a brief overview of some of the most notable ones is presented as follows:

a) ***Dynamic multi-swarm PSO [76]***: This is arguably the first highly notable multi-swarm PSO approach. It divides the population into many small dynamic swarms which are evolved in a similar fashion; these swarms are regrouped frequently thereby exchanging information among the swarms. This multi-swarm PSO exhibited better performance than several PSO variants on a set of shifted rotated benchmark functions.

b) ***The Socio-Cognitively Inspired PSO [77]***: This approach divides the PSO swarm into sub-swarms, called species because they have unique evolution mechanisms. The particles get inspired by the global and the local optima, but they also share their knowledge of optimal locations with neighboring particles belonging to other species. Experiments were conducted with various proportions of different species in the population to find the best setting.

c) ***HCLDMS-PSO [78]***: The heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators (HCLDMS-PSO) modifies the traditional dynamic multi-swarm PSO [76] by adding a comprehensive learning (CL) strategy, where the global optimal experience of the whole population is conducted to generate an exploitation sub-population exemplar.

d) ***HIDMS-PSO [79]***: The Heterogeneous Improved Dynamic Multi-Swarm PSO (HIDMS-PSO) is also an extension of the traditional dynamic multi-swarm PSO [76]. The population is initially divided into two sub-populations, with the first one being further divided into sub-swarms. The particles in these sub-swarms are guided heterogeneously. On the other hand, the second sub-population is guided homogeneously by using the classical PSO update mechanism.

e) ***DMS-GPSO [80]***: The dynamic multi-swarm global particle swarm optimization (DMS-GPSO) segments the evolutionary process into an initial stage and a final stage. In the initial stage, the population is divided into a global sub-swarm and multiple dynamic sub-swarms. The global sub-swarm focuses on exploitation, guided by the globally optimal particle. On the other hand, multiple sub-swarms focus on exploration, guided by the best particle in the neighborhood. In the final stage, the elite particles stored in an archive are combined with the DMS sub-swarms to form a single population with the aim of improving exploitation capabilities.

f) ***DMS-PSO-GD [81]***: The Dynamic Multi-Swarm Particle Swarm Optimization with Global Detection Mechanism (DMS-PSO-GD) has some similarities to the DMS-GPSO [80] in the sense that it also divides the population into two kinds of sub-swarms: several same-sized dynamic sub-swarms and a global sub-swarm. This algorithm, however, uses the variances and average fitness values of dynamic sub-swarms to measure the distribution of the particles, in order to detect the dominant and the optimal particle.

g) ***PSOMAS [82]***: In the Particle Swarm Optimization with Multiple Adaptive Sub-swarms (PSOMAS) algorithm, each sub-swarm is evolved by a completely different variant of the single swarm PSO algorithm, such as Comprehensive Learning PSO [83] and Cooperative PSO [84]. This work also uses an adaptive strategy to reduce usage of computational resources.

h) ***Chain and Hypercube Communication Topologies for multiswarm-PSOs [85]***: The chain and hypercube topologies have been proposed with the aim of limiting communication between swarms in order to increase per-swarm exploration over different parts of the search space.

A comparison of these techniques with the simple cross-over strategy showed that the chain topology exhibited a better error performance. The computational complexities of these techniques is, however, not discussed.

4.4 Multi-objective Workflow Scheduling Based on MS-PSO

Scientific workflow scheduling is a computationally complex exercise. Therefore, the multi-swarm PSO optimization algorithm tailored for this problem must be simple and characterized by a low computational complexity. In light of this understanding, this work adopts the Socio-Cognitively Inspired PSO [77] due to its apparent simplicity. Furthermore, in this approach, the swarms or species are evolved differently from each other. The particles' position and velocity are updated by the different rules specific to a swarm. They, however, exchange information with the neighboring swarms. This behavior is very intuitive from natural point of view because in the real world, a variety of different species of a particular animal usually exist in an ecosystem simultaneously.

The competition for territory and survival creates the need for inter-swarm communication. Therefore, the evolution of the individuals is affected by other swarms' rules thereby promoting information sharing among swarms and cooperative development of the all species. In the proposed MS-PSO algorithm, each swarm is initialized with particles that belong to a particular type of species. The species in each swarm has its own algorithm for calculating the velocity. In stead of using the neighborhood's best position in the third term as in [77], entire swarm's best position is used. Therefore, at every k-th iteration, the elements of velocity $\mathbf{v}_i^s$ of the i-th particle in the s-th swarm are updated by using

$$\begin{aligned} \mathbf{v}_{i,h}^s = \omega \mathbf{v}_{i,h}^s + A(\text{pBest}_{i,h}^s - \mathbf{x}_{i,h}) \\ + B(\text{sBest}_h^s - \mathbf{x}_{i,h}) + C(\text{gBest}_h - \mathbf{x}_{i,h}), \end{aligned} \tag{4.5}$$

where $\mathbf{pBest}_i^s$ is known as the personal best position for the i-th particle in the s-th swarm; $\mathbf{sBest}^s$ is the best position in the s-th swarm and **gBest** is the global best position found across all swarms. A, B and C are the learning factor coefficients in the range [0,1] which are different for each swarm according to its class of species. The elements of position $\mathbf{x}_i^s$ of the i-th particle are updated by using

$$\mathbf{x}_{i,h}^s = \mathbf{x}_{i,h}^s + \mathbf{v}_{i,h}^s, \tag{4.6}$$

The five types of species, considered in this work, are presented as follows:

- *Normal Species:* This species represents the canonical PSO, where the particle's decisions are affected by the particle's best solution and a swarm's best solution, with each given the same weighted coefficient.
- *Local and Global Species:* This species is influenced only by its own best and global best position for all swarms.
- *Swarm only Species:* This species is influenced only by the swarm's best position.
- *Global only species:* This species is influenced only by the global best position for all swarms.
- *Random species:* This species employs randomness, where all three parameters in the PSO velocity function (Eq. (4.5)) are taken into consideration. For each iteration, the coefficient weights of the parameters are uniformly distributed random number in the range (0,1).

Table 4.1 shows the parameters for the five species. Once the position of a particle in the swarm has been updated according to equations (4.5) and (4.6), it's fitness value is determined according to equation (4.3). If the new fitness value is better than the previous one, then the respective best values are updated by the current values.

Table 4.1: Settings of the cognitive and social learning coefficients for different species

Species Type	Particle (A)	Swarm (B)	All Swarms (C)
Normal	1	1	0
Local & Global	1	0	1
Swarm only	0	1	0
Global only	0	0	1
Random	random	random	random

Algorithm 4 illustrates the Multi-Swarm PSO-based optimization process. The input to the algorithm is the workflow wf and all the available cloud-fog VMs, while the output is the task-VM mapping schedule - **gBest** and $F(\mathbf{gBest})$. Parameters N, S and G denote the number of particles, the number of swarms, and the number of generations respectively.

The algorithm starts with the initialization of the values of the following parameters: N, S, A, B,

Algorithm 4: The Proposed Multi-Swarm PSO-based Algorithm

Input: Workflow wf composed of t tasks and all available cloud-fog VMs;
Output: Task-VM mapping schedule - **gBest** and $F(\mathbf{gBest})$;
Initialize N, S, A, B, C, ω, and G; Randomly generate N particles; Divide the N particles into S subswarms of equal sizes;
$F(\mathbf{gBest}) \leftarrow 0$;
for $s \leftarrow 1, S$ **do**
    $F(\mathbf{sBest}^s) \leftarrow 0$;
    for $i \leftarrow 1, N/S$ **do**
        Invoke the Fogworkflowsim workflow scheduler; Compute the fitness function value, $F(\mathbf{x}_i^s)$, for particle i in swarm s, by using equation (4.3) in Section IV;
        $\mathbf{pBest}_i^s \leftarrow \mathbf{x}_i^s$; $F(\mathbf{pBest}_i^s) \leftarrow F(\mathbf{x}_i^s)$;
        if $F(\mathbf{x}_i^s) > F(\mathbf{sBest}^s)$ **then**
            $\mathbf{sBest}^s \leftarrow \mathbf{x}_i^s$; $F(\mathbf{sBest}^s) \leftarrow F(\mathbf{x}_i^s)$;
        if $F(\mathbf{x}_i^s) > F(\mathbf{gBest})$ **then**
            $\mathbf{gBest} \leftarrow \mathbf{x}_i^s$; $F(\mathbf{gBest}) \leftarrow F(\mathbf{x}_i^s)$;
$k \leftarrow 0$;
while $k \leq G$ **do**
    for $s \leftarrow 1, S$ **do**
        for $i \leftarrow 1, N/S$ **do**
            (1) Update $\mathbf{v}_i^s$ and $\mathbf{x}_i^s$ by using equations (4.5) and (4.6), and species-specific parameters from Table 4.1;
            (2) Invoke the Fogworkflowsim workflow scheduler;
            (3) Compute the fitness function value, $F(\mathbf{x}_i^s)$, by using the Weighted Sum Objective Function (equation (4.3) in Section IV);
            if $F(\mathbf{x}_i^s) > F(\mathbf{pBest}_i^s)$ **then**
                $\mathbf{pBest}_i^s \leftarrow \mathbf{x}_i^s$; $F(\mathbf{pBest}_i^s) \leftarrow F(\mathbf{x}_i^s)$;
            if $F(\mathbf{x}_i^s) > F(\mathbf{sBest}^s)$ **then**
                $\mathbf{sBest}^s \leftarrow \mathbf{x}_i^s$; $F(\mathbf{sBest}^s) \leftarrow F(\mathbf{x}_i^s)$;
            if $F(\mathbf{x}_i^s) > F(\mathbf{gBest})$ **then**
                $\mathbf{gBest} \leftarrow \mathbf{x}_i^s$; $F(\mathbf{gBest}) \leftarrow F(\mathbf{x}_i^s)$;
    $k \leftarrow k + 1$;

C, ω, and G. In the next step, N particles are created and divided into S swarms of equal sizes and the fitness value of the best global position $F(\mathbf{gBest})$ is initialized to 0.

Between line 7 and line 19, the fitness values of the particles in each swarm are determined according to equation (4.3), by running the respective workflow scheduling simulation on Fogworkflowsim [28]. The $\mathbf{pBest}_i^s$, $\mathbf{sBest}^s$ and $\mathbf{gBest}$ values are determined among all the swarms. Once this process is completed, each of the swarms/species is evolved by swarm/species-specific parameter settings presented in Table 4.1 in an iterative process until G generations are reached. This happens from 21 to line 36. At every iteration, the computation goes through two loops: 1) the outer loop for the swarms; 2) the inner loop for the individual particles in a particular swarm. Within the inner loop, there are three steps:

1. The update of the particle's velocity and position by using equations (4.5) and (4.6), and species-specific parameters from Table 4.1.
2. The invokation of the Fogworkflowsim workflow scheduler to determine the fitness of the particle determined in the first step.
3. The calculation of the fitness function value, $F(\mathbf{x}_i^s)$, by using the Weighted Sum Objective Function (equation (4.3), in Section IV).

From line 27 to line 35, the newly determined fitness value $F(\mathbf{x}_i^s)$ for a particle i in swarm s is compared with the three best fitness values associated with the particle at personal, swarm, and global levels $F(\mathbf{pBest}_i^s)$, $F(\mathbf{sBest}^s)$ and $F(\mathbf{gBest})$ respectively). The values for $\mathbf{pBest}_i^s$, $F(\mathbf{pBest}_i^s)$, $\mathbf{sBest}^s$, $F(\mathbf{sBest}^s)$, $\mathbf{gBest}$, and $F(\mathbf{gBest})$ are updated, whenever better values are found in each case.

4.5 Performance Evaluation

This section begins by describing simulation environment setup on the FogWorkflowSim Toolkit. This is followed by the experimental results and discussion of the comparative evaluation of MS-PSO, GA, PSO, GA-PSO and DE as scheduling algorithms for scientific workflows is conducted with makespan, cost, load balancing, and energy as performance metrics. The number of tasks has been increased to a maximum of 500 in each case.

4.5.1 Simulation Environment

The overall setup of the experiments and the simulation environment is described here. The simulations are conducted using FogWorkflowSim [28]. The performance of the proposed MS-PSO is evaluated on five scientific workflow types, as described previously and the results are compared with four other population-based algorithms, namely, standard GA, PSO, DE and GA-PSO proposed in [14]. The final parameter settings for all the experiments are based on preliminary runs. The population size = 50 for each algorithm.

Table 4.2: Parameter Settings Of Cloud-Fog Environment

Parameters	**End Device**	**Fog VM**	**Cloud VM**
Processing rate (MIPS)	500	1000	2000
Task execution cost ($)	0	0.48	0.96
Communication cost ($)	0	0.01	0.02
Working power (mW)	200	700	1700
Idle power (mW)	50	200	1200
Uplink bandwidth (Mbps)	800	500	300
Downlink bandwidth (Mbps)	1000	800	500

The MS-PSO learning factors A, B and C are set according to Table 4.1, with $S = 5$ and each containing 10 particles. The inertia weight $\omega = 1$. The classical PSO learning factors $C_1 = C_2 = 2$, with inertia weight $\omega = 1$. The GA algorithm's crossover and mutation rates are 0.8 and 0.1, respectively. The DE crossover probability = 0.9 and the differential weight = 0.8. The number of iterations for all algorithms is 100. The simulations are executed 10 times for each workflow and task volume to get the average performance of the algorithms. The simulator is setup with one end device, six fog VMs and ten cloud VMs. The characteristics for each VM on the three cloud-fog layers along with the environment settings are shown in Table 4.2.

4.5.2 Simulation Results and Discussion

The performance of the proposed algorithm is compared with the canonical PSO and the GA, DE and GA-PSO, which are the three other population-search based algorithms for cloud based workflow scheduling. All the algorithms are executed on the same simulation environment setup.

The respective fitness evolution graphs and run-times for each algorithm are recorded and presented. Finally, the comparison of the results is presented in terms of the following performance metrics: makespan, cost, energy and load balancing on the cloud and fog respectively.

Fig. 4.1 shows the evolution of the fitness value for each scientific workflow with 300 tasks and the overall quality of the task schedule produced by the algorithms considering all the performance metrics. For the Montage workflow, as shown in Fig. 4.1(a), the MS-PSO approach closely follows the DE approach, which achieves the best fitness value. On the other hand the canonical PSO approach exhibits the worst performance closely followed by the GA approach. In the case of Cybershake (Fig. 4.1(b)), the rest of the algorithms apart from the PSO approach achieve the similar fitness value. The GA and the DE approaches seem to converge faster than the rest of the algorithms. The Epigenomics workflow, in Fig. 4.1(c), exhibits a similar pattern to the Montage workflow (see Fig. 4.1(a)) in the sense that MS-PSO and DE achieve the lowest fitness values, while GA and PSO achieves the highest values. In Fig. 4.1(d), the MS-PSO again follows the DE, GA-PSO and GA algorithms, which are the best performing approaches, closely. The SIPHT workflow, Fig. 4.1(e), shows a similar pattern. In all these cases, all algorithms seem to converge by the 30th iteration.

From the results in Fig. 4.1, it can clearly be seen that the final fitness value of the proposed MS-PSO is much smaller than the canonical PSO and very similar to the other algorithms. This indicates that the MS-PSO has the ability to find near-optimal solutions, while the canonical PSO easily falls into a local optima. The MS-PSO benefits from the multi-species approach where the particle's velocity update in each swarm is calculated differently, enabling a more exploratory global search with the capability to easily jump local optima.

Fig. 4.2 - Fig. 4.6 shows the average performance metric results of the task schedules realized by the five algorithms for five workflow types under 100, 300, and 500 tasks. Fig. 4.2 shows that the Montage workflow achieves the shortest makespan of the five workflows. At 100 tasks, all the algorithms exhibit a similar performance. At 300 tasks, PSO is the worst while DE is the best approach. The rest of the algorithms lie in between. At 500 tasks, DE joins PSO as the worst performing algorithms, while GA and GA-PSO achieve the best performance. The MS-PSO shows stability and reliability.

In the Cybershake workflow, the PSO algorithm is the worst at 100 and 300 tasks, and it is joined

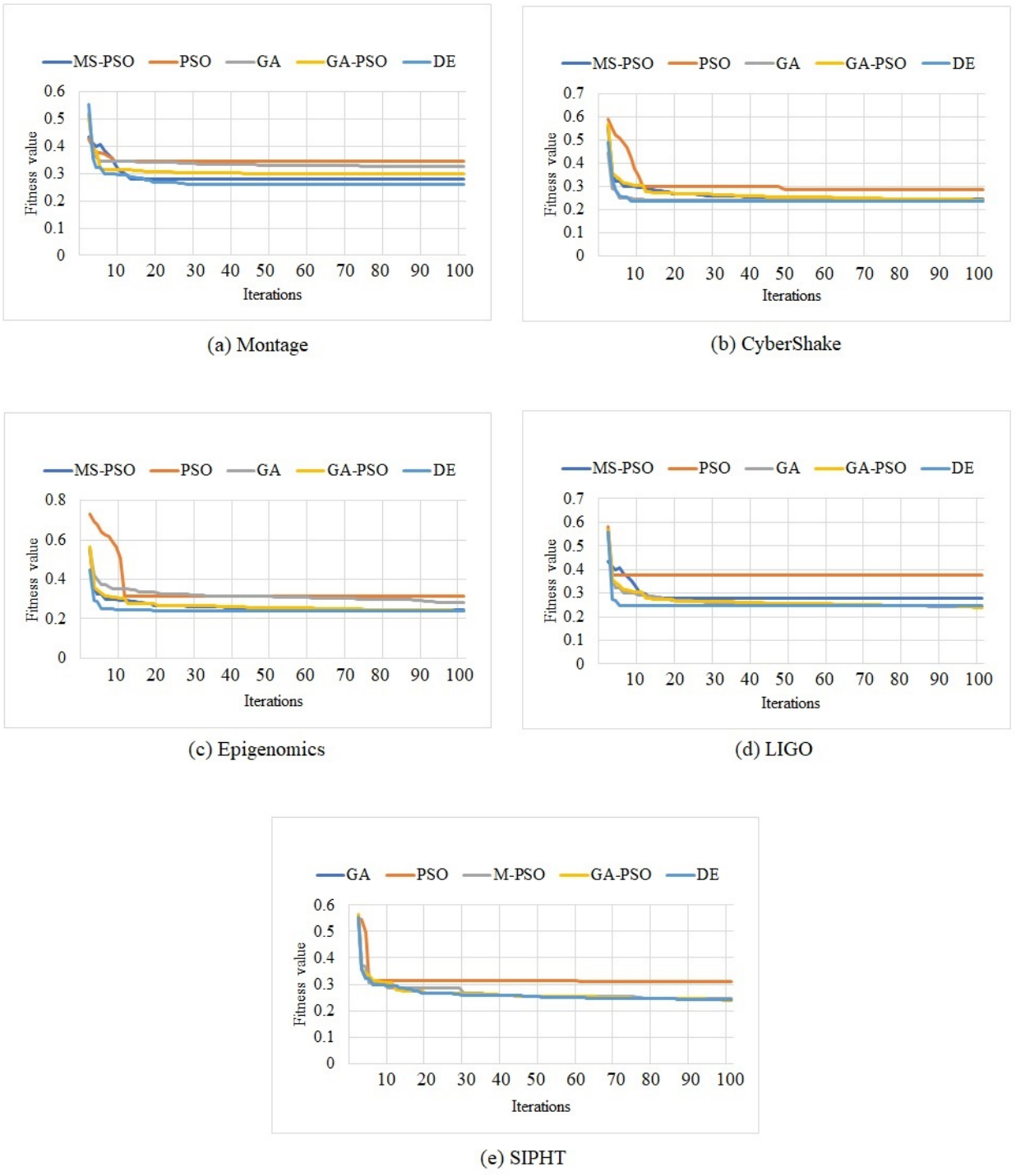
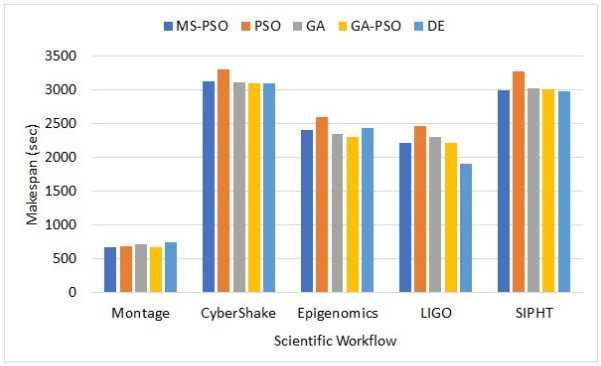
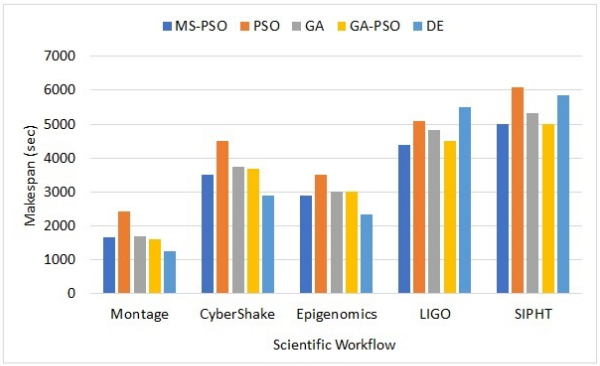


Figure 4.1: Comparison of fitness value evolution

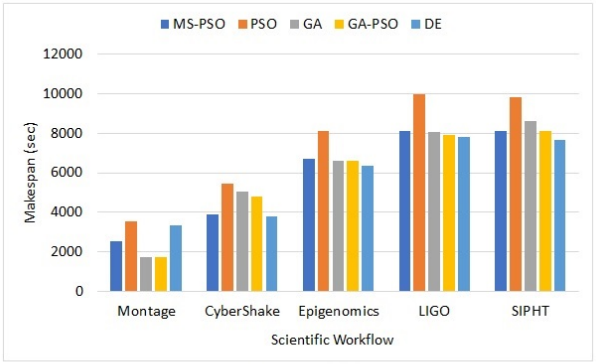
by PSO, GA-PSO and GA at 500 tasks. At the other end, the rest of the algorithms perform equally at 100 tasks. The DE is the best at 300 tasks and it is joined by the MS-PSO at 500 tasks. Again, the proposed MS-PSO exhibits a better performance than the canonical PSO and it is more



(a) 100 tasks



(b) 300 tasks



(c) 500 tasks

Figure 4.2: Comparison of makespan for the different algorithms

stable than the rest of the algorithms. In the Epigenomics workflow, performance is more or less the same at 100 tasks. At 300 tasks, PSO is the worst, while DE is the best. At 500 tasks, PSO is still the worst while the rest of the algorithms perform more or less in a similar manner.

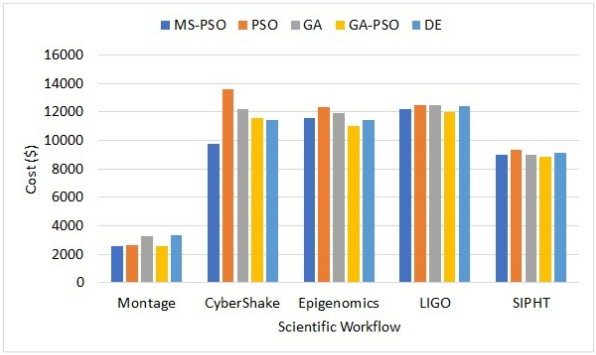
In the LIGO workflow, DE is the best, while the rest of the approaches exhibit similar performance, at 100 tasks. At 300 tasks, the DE is the worst, while the MS-PSO is the best. At 500 tasks, PSO is the worst, while the rest of the algorithms exhibit similar performance. Finally, in the SIPHT workflow, PSO has worst makespan at 100 tasks, while the rest of the algorithms exhibit a similar performance. At 300 tasks, PSO is joined by DE with the worst makespan. The MS-PSO and GA-PSO achieve the best makespan values. At 500 tasks, PSO still performs poorly, while the rest of the algorithms perform similarly.

The proposed MS-PSO approach clearly outperforms the canonical PSO approach, while competing fairly well with the other approaches. It comes out on top in several instances, while performing

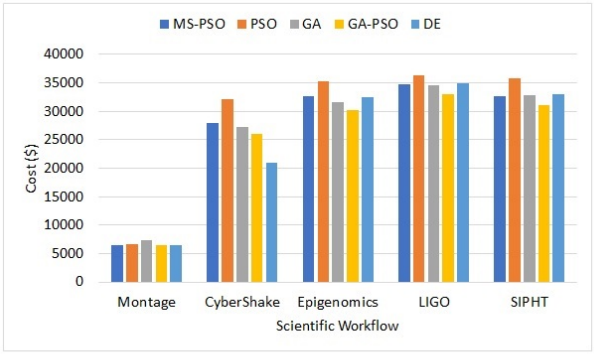
closer to the top in many cases. Amongst the competitors, DE and GA-PSO produce the best results on several instances, but they also register the worst results in a number of cases. The proposed MS-PSO approach is, therefore, stable and more reliable. This can be attributed to the use of different velocity update mechanisms in the sub-swarms. This helps to guarantee efficiency in times of when some sub-swarms become unstable or are stuck in the local minima. The MS-PSO generally achieves better results than the GA-PSO. This suggests that the parallel approach of combining algorithms might probably generate good results, as opposed to the serial approach in the GA-PSO approach.

Fig. 4.3 illustrates a comparative evaluation based on cost. The Montage workflow yields the lowest makespan and as a result, it also registers the lowest cost. At 100 tasks, MS-PSO, PSO and GA-PSO achieve the lowest cost, while GA and DE are the worst in terms of cost. There is, however, similar performance among the algorithms when the tasks are increased to 300 and 500. For the Cybershake workflow, MS-PSO has the lowest cost at 100 tasks, while PSO has the highest cost. When the number of tasks is increased to 300, DE yields the lowest cost, while PSO still has the highest cost. The proposed MS-PSO ranks between the two extremes. At 500 tasks, PSO remains with the highest cost, while the rest of the algorithms have similar cost values. For the Epigenomics workflow, the performance is more or less the same for all algorithms for all tasks. The same applies to the LIGO workflow. For the SIPHT workflow, all algorithms register similar costs for 100 and 500 tasks. For 300 tasks, only the PSO yields a high cost while the rest of the algorithms have similar costs. On the overall cost analysis, the PSO gives the highest cost, while the proposed MS-PSO still competes fairly well with the rest of the algorithms. MS-PSO and DE come out on top once.

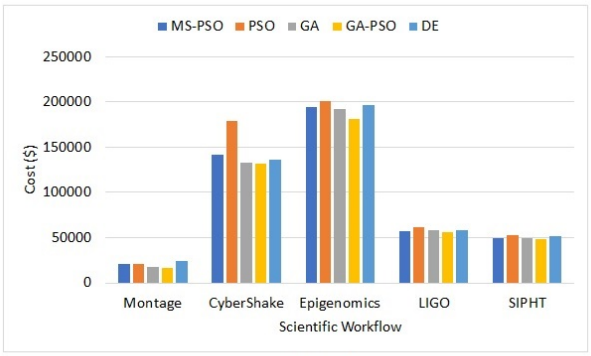
Fig. 4.4 illustrates a comparative evaluation based on energy consumption. As expected, the Montage workflow consumes the lowest amount of energy under all tasks. At 100 tasks, all algorithms have similar energy consumption, while PSO and the DE are the worst and the best performing algorithms when the number of tasks is increased to 300. At 500 tasks, the DE becomes the worst algorithm while GA and GA-PSO are the best; MS-PSO and PSO exhibit similar performance. For the Cybershake workflow, MS-PSO has the lowest energy consumption at 100 tasks, while the rest of the algorithms exhibit similar performance. At 300 tasks, DE yields the lowest energy consumption, while PSO still has the highest energy consumption. The proposed MS-PSO ranks between the two extremes. At 500 tasks, MS-PSO gives the best performance, closely followed by



(a) 100 tasks



(b) 300 tasks



(c) 500 tasks

Figure 4.3: Comparison of cost for the different algorithms

DE. The rest of the algorithms have similar energy values.

For the Epigenomics workflow, PSO consumes the highest amount of energy while the rest of the algorithms show similar energy consumption for 100 and 500 tasks. At 300 tasks, the trend is the same, but DE yields slightly better performance than the other algorithms. For the LIGO workflow, DE slightly outperforms the rest of the algorithms for 100 tasks. On the other hand, PSO consumes the highest energy, while the rest of the algorithms yield similar energy consumption values for 300 and 500 tasks. For the SIPHT workflow, all algorithms register similar costs for 100 and 500 tasks. For 300 tasks, only the PSO yields the highest energy consumption while DE has a slightly lower energy consumption than the other three competitors.

On the overall energy analysis, the proposed MS-PSO and DE yield the lowest energy consumption on several instances. Ironically, DE also registers the highest energy consumption on one instance. This further confirms the fact that the DE algorithm is unstable and unreliable. This stems from the fact that it uses one homogeneous population. Therefore, the possibility of premature convergence

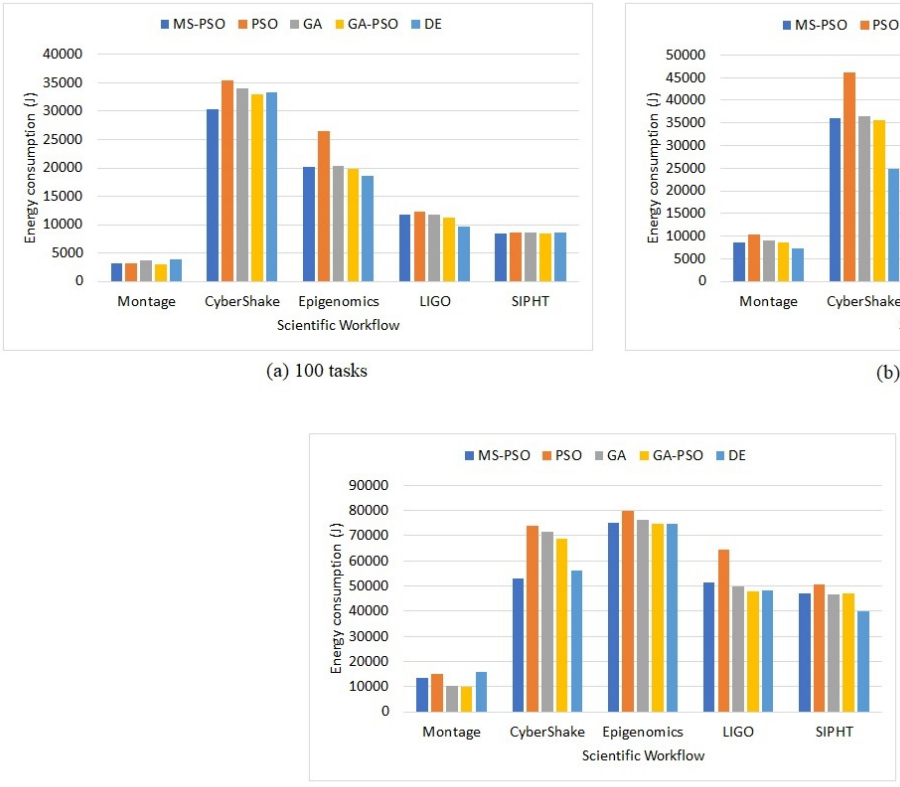


Figure 4.4: Comparison of energy consumption for the different algorithms

is always there.

Fig. 4.5 illustrates a comparative evaluation based on load balancing on the cloud resources. For the Montage workflow, all algorithms register similar load balancing performance for 100 tasks. At 300 tasks, PSO and the DE are the worst and the best performing algorithms, respectively. On the other hand, GA performs best at 500 tasks, with PSO, GA-PSO and DE exhibiting the worst performance. MS-PSO's load balancing performance is in between the two extremes. For the Cybershake workflow, MS-PSO and PSO have the best and worst performance for 100 tasks, respectively. At 300 tasks, DE and PSO have the best and worst performance, while at 500 tasks DE and MS-PSO have the best performance. For the Epigenomics workflow, PSO has the worst performance while the rest of the algorithms show similar performance for 100 tasks. The DE yields better performance than the other algorithms for 300 and 500 tasks. For the LIGO workflow, all algorithms except PSO, exhibit similar performance for all tasks. For the SIPHT workflow, PSO and MS-PSO have the worst performance while the rest of the algorithms register similar costs for

100 tasks. For 300 tasks, PSO and DE have the best and worst performance, while at 500 tasks, only PSO performs badly.

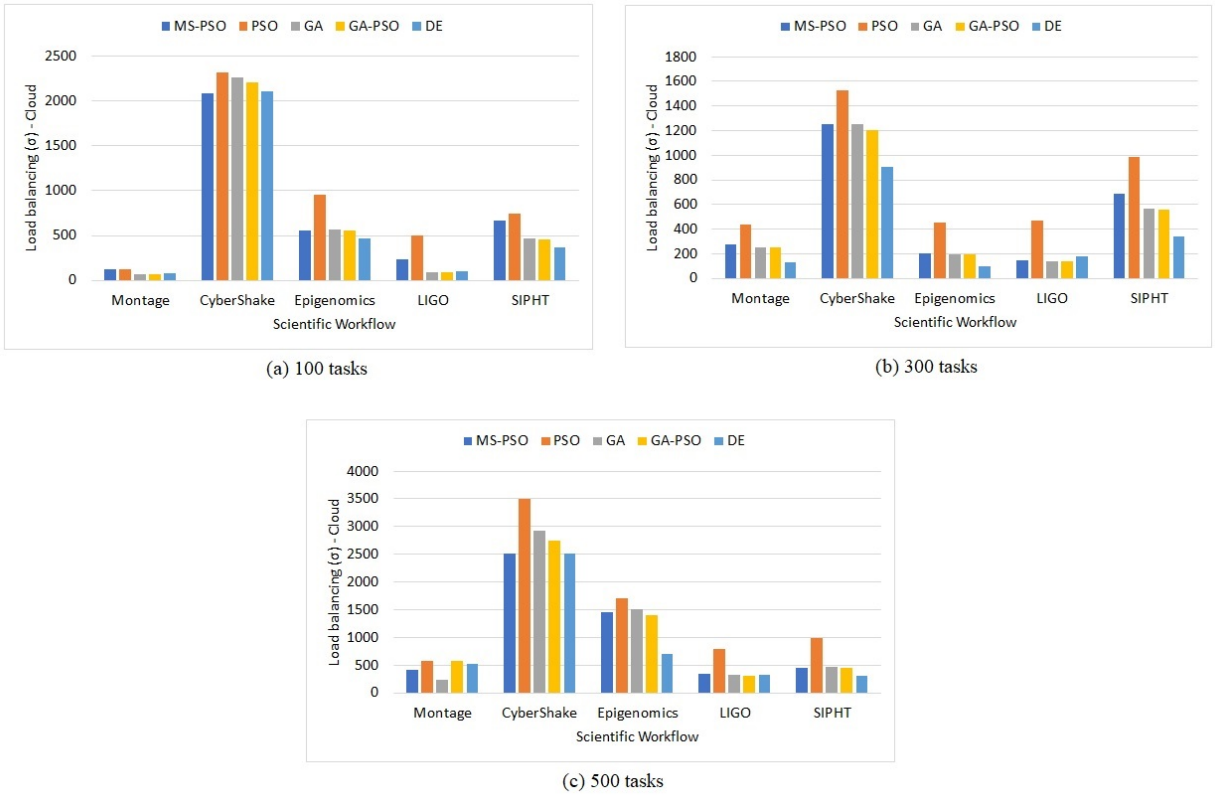
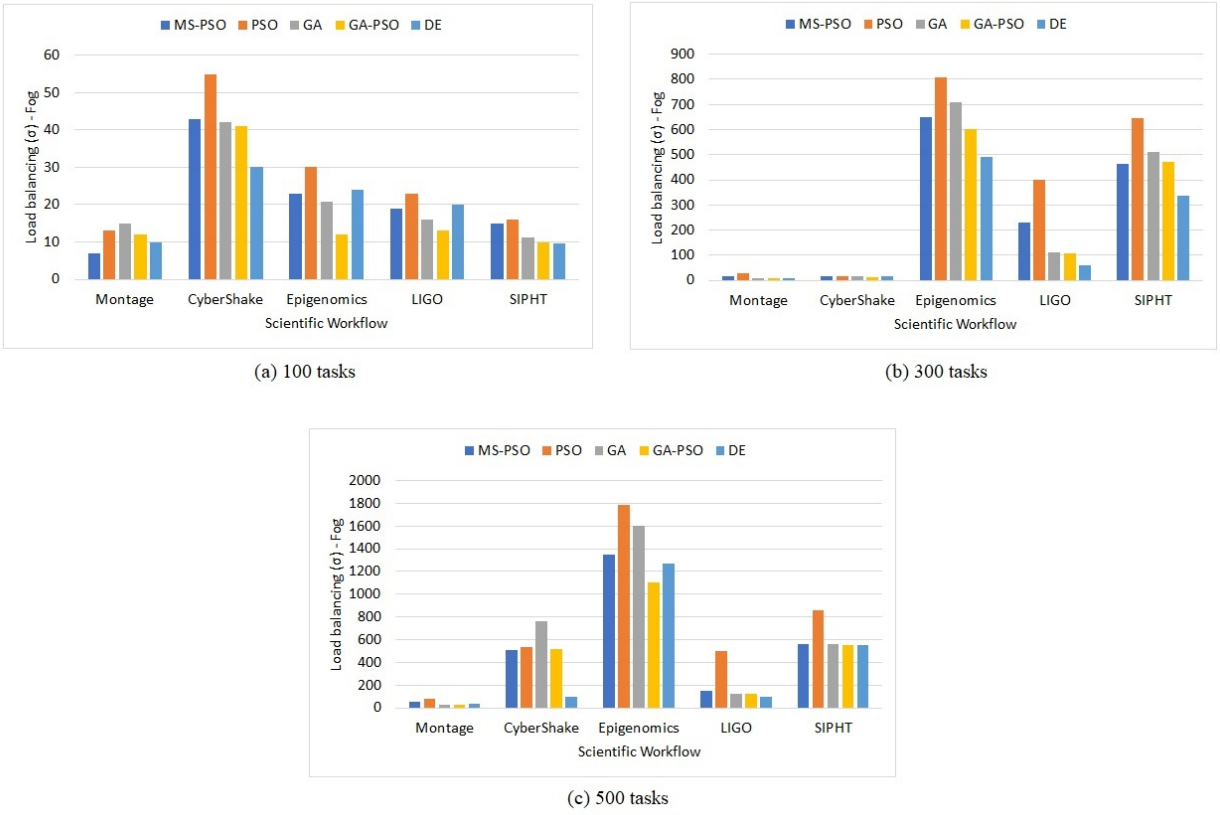


Figure 4.5: Comparison of load balancing on the cloud for the different algorithms

In the overall analysis on load balancing on the cloud, DE yields the best performance on most instances while MS-PSO ranks second. Nevertheless DE yields the worst performance on one instance. This further confirms the problem of instability as previously explained.

Fig. 4.6 illustrates a comparative evaluation based on load balancing on the fog resources. For the Montage workflow, when the number of workflow tasks is set to 100, MS-PSO achieves the best load balancing performance, with the GA giving the worst performance. PSO performs slightly poorly, while the rest of the algorithms perform similarly for 300 and 500 tasks. For the Cybershake workflow, DE and PSO have the best and worst performance for 100 tasks. At 300 tasks, all algorithms have similar performance, while DE and GA have the best and worst performance for 500 tasks. For the Epigenomics workflow, GA-PSO and PSO have the best and worst performance for 100 and 500 tasks. On the other hand, DE and PSO have the best and worst performance

for 300 tasks. For the LIGO workflow, GA-PSO and PSO have the best and worst performance for 100 tasks. At 300 tasks, DE slightly outperforms GA-PSO to emerge as the best performing algorithm, while ony PSO performs badly at 500 tasks. For the SIPHT workflow, PSO has the worst performance; the MS-PSO follows PSO closely, while the rest of the algorithms register similar costs for 100 tasks. For 300 tasks, PSO and DE have the best and worst performance, while at 500 tasks, just like in the case of cloud-based load balancing, only PSO performs badly.



(a) 100 tasks

(b) 300 tasks

(c) 500 tasks

Figure 4.6: Comparison of load balancing on the fog for the different algorithms

In the overall analysis on load balancing on the fog devices, DE yields the best performance on most instances while GA-PSO and MS-PSO rank second and third respectively.

Having compared the algorithms based on various performance metrics, it is also important to consider the computational times of the algorithms. Therefore, Table 4.3 shows a comparison of the average execution times of the algorithms for the five scientific workflows using different number of tasks. GA and DE have the worst execution times. On the other hand, the approaches that incorporate PSO have much better execution times, with the canonical PSO clearly outperforming

Table 4.3: The average execution time of the algorithms

Workflow	Number of tasks	Algorithm runtime /sec				
		MS-PSO	PSO	GA	GA-PSO	DE
Montage	100	14.961	**10.656**	25.247	16.523	25.424
	300	56.456	**51.279**	76.009	60.241	77.302
	500	158.644	**153.408**	192.513	165.19	194.139
CyberShake	100	14.422	**10.395**	24.174	15.908	24.266
	300	48.333	**42.252**	61.028	49.897	69.346
	500	130.358	**115.924**	145.265	133.29	144.698
Epigenomics	100	13.061	**10.819**	24.107	15.344	24.37
	300	56.405	**51.78**	71.793	58.782	76.651
	500	178.086	**171.654**	209.732	184.292	207.713
LIGO	100	13.233	**10.407**	35.116	17.892	36.117
	300	59.88	**55.063**	76.776	64.797	79.474
	500	170.164	**161.873**	196.134	174.329	201.607
SIPHT	100	109.573	**104.274**	125.372	112.439	120.545
	300	368.48	**362.949**	392.557	373.915	390.109
	500	740.411	**730.311**	777.945	747.912	782.624

the rest of the algorithms in all cases. This highlights the computational and implementation simplicity of the canonical PSO technique. The proposed MS-PSO algorithm inherits those attributes, albeit at a slightly increased computational cost due to the addition of the third term that captures the contribution of the best position for each swarm and determining each swarms' best performing particle for every iteration. However, the canonical PSO produced the lowest quality of solutions as shown in Fig. 4.1 - Fig. 4.6. It is noteworthy that as the workflow size increases, the MS-PSO offers a good trade-off between algorithm execution time and quality of solutions produced. This is crucial especially for large-scale scientific workflow applications where long running times can have a significant impact on financial cost and energy when generating task schedules for a cloud-fog environment.

These results show that the canonical PSO algorithm performs the worst on all the workflow instances due to PSO's well-known problem of premature convergence, which leads it to getting trapped in a local optima. The MS-PSO algorithm, on the other hand, outperforms the canonical PSO for the five scientific workflows and competes fairly well with the other algorithms. The MS-PSO also exhibits stability; it has rarely offered the worst performance. This demonstrates the high reliability of the MS-PSO approach. The MS-PSO benefits from the variety of cognitive and social learning coefficients of the different swarms when determining the velocity for each particle, hence the algorithm is able to escape local optima. The results are affected by the characteristics of the workflow instances. When the number of tasks for the respective workflow instance is higher, the metric values are also generally higher as the run times and data sizes of the workflow DAG are significantly higher.

4.6 Chapter Summary

This chapter presented a multi-swarm Particle Swarm Optimisation (MS-PSO) approach for scientific workflow scheduling in cloud-fog environments. This approach is motivated by the problem of premature convergence in the PSO technique. This chapter also developed the concept of load balancing, both for the fog and the cloud resources, and incorporated it in the weighted multi-objective mechanism. The state-of-the-art multi-swarm PSO algorithms have also been reviewed with the aim of determining the best approach for implementing a multi-swarm PSO technique for scientific workflow scheduling. The resulting MS-PSO uses multiple swarms of different species of particles to

improve exploration of the search space and avoid premature convergence. Finally, simulations are performed with five scientific workflows using the FogWorkflowSim tool. The proposed algorithm is compared to the canonical PSO, GA, DE and GA-PSO. The following performance metrics are used: makespan, cost, energy consumption, and load balancing on the fog as well as on the cloud resources. Results show that the proposed MS-PSO generally outperforms the canonical PSO on all scientific workflows and under all performance metrics. The proposed MS-PSO generally performs better than GA and GA-PSO. It competes fairly well against DE and more importantly it is more stable and reliable than DE. In terms of computational complexity, the proposed MS-PSO only ranks second to PSO, while DE has the worst execution time. These results show that the proposed MS-PSO technique is exhibiting much better overall performance compared to the competitors. The next chapter presents the application of a DE variant, known as SHADE, to scientific workflow scheduling.

Chapter 5

Success-History-based Differential Evolution (SHADE) for Scientific Workflow Scheduling in Cloud-Fog Environments

5.1 Introduction

This chapter presents the application of the Success-History-based Adaptive Differential Evolution (SHADE) algorithm to the problem of scientific workflow scheduling in a cloud-fog environment. The weighted sum objective function that incorporates makespan, cost and energy as described in Chapter 3 is used. The simulations compare the SHADE mechanism with the canonical Differential Evolution (DE). The following scientific workflows are used: Montage, CyberShake, SIPHT, LIGO and Epigenomics. Section 5.2 presents the details of the SHADE variant used in this work. Section 5.3 presents the simulation environment setup with the algorithm parameters. Then Section 5.4 discusses the simulation results. Finally, Section 5.5 concludes the chapter.

5.2 Description of SHADE variant - DISH

This work uses the SHADE variant proposed in [47]. This algorithm introduced the distance based parameter adaptation to the SHADE algorithm in order to address the premature convergence of SHADE–based algorithms. As a result, the algorithm utilizes a longer exploration search, especially in higher dimensional search spaces. Furthermore, the modification proposed in [47] to the original scaling factor and crossover rate parameters in SHADE rewards the exploration capabilities more. In this section, the major aspects of the SHADE algorithm are presented before specifically presenting the distance parameter adaptation mechanism, described as DISH (**DI**stance Based Parameter Adaptation for **S**uccess-**H**istory based Differential Evolution).

5.2.1 SHADE

In order to get rid of the need to fine-tune the control parameters F and CR, the SHADE algorithm [86, 87] was proposed. SHADE maintains two historical memories $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ for successful scaling factor and crossover rate values with their update mechanism. The various steps of the SHADE algorithm are presented as follows.

Initialization

The initial population is generated randomly just like in the case of the canonical DE [44]. In addition, historical memories are preset to 0.5 for both scaling factor and crossover rate parameters. Furthermore, an external archive of inferior solutions $\mathbf{A}$ has to be initialized to $\emptyset$. Once the initialization process is completed, the rest of the steps (mutation, crossover, selection, historical memory updates) are taken until the stopping criterion is reached.

Mutation

The SHADE algorithm replaces the mutation in Eq. 2.3, with

$$\mathbf{v}_{i,j} = \mathbf{x}_{i,j} + F_i(\mathbf{x}_{\text{pbest},j}\mathbf{x}_{i,j}) + F_i(\mathbf{x}_{r1,j}\mathbf{x}_{r2,j}), \tag{5.1}$$

where pbest is an index of one of the $p \times 100\%$ best individuals and pbest $\neq r_1 \neq r_2$. The scaling factor value F_i is given by

$$F_i = \mathcal{C}(\mathbf{M}_{F,r}, 0.1), \tag{5.2}$$

where $\mathbf{M}_{F,r}$ is a randomly selected value from historical data of $\mathbf{M}_F$ values. This happens because index r is randomly generated from the range $[1, \text{Hm}]$), where Hm is the memory size of historical data. Function $\mathcal{C}()$ is the Cauchy distribution, which is implemented in such a way that $0.0 \leq F_i \leq 1.0$.

Crossover and Selection

The crossover operation is similar to the canonical DE operation in Section 2.4, in Chapter 2. The only difference is in how the CR is determined. In the SHADE approach, the historical data saved in $\mathbf{M}_{CR}$ is utilized by using the Gaussian distribution illustrated by

$$CR_i = \mathcal{N}(\mathbf{M}_{CR,r}, 0.1), \tag{5.3}$$

where $\mathbf{M}_{CR,r}$ is a randomly selected value from $\mathbf{M}_{CR}$ and $\text{CR}_i \in [0, 1]$. Once the crossover has taken place, the selection process is conducted in the same manner as in the canonical DE.

5.2.2 Historical Memory Updates

The historical memories $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ keep successful values F and CR from the mutation and crossover processes; values that produce a trial individual better than the original individual are deemed to be successful. These F and CR values are stored in $\mathbf{S}_F$ and $\mathbf{S}_{CR}$ respectively. Furthermore, the corresponding goal function improvements $S_{\Delta f_i}$, where $\Delta f_i = |f(u_i) - f(x_i)|$, are calculated. The memory cells are updated using the values calculated with weighted Lehmer mean [88]:

$$mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j S_j^2}{\sum_{j=1}^{|S|} w_j S_j}, \tag{5.4}$$

where $w_j = \frac{S_{\Delta f_i}}{\sum_{k=1}^{|S|} S_{\Delta f_k}}$ and $S = \mathbf{S}_F$ or $S = \mathbf{S}_{CR}$. The new values for $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ values after generation g for one of the memory cells h are given by

$$\mathbf{M}_{F,h}^{g+1} = 0.5(\mathbf{M}_{F,h}^{g} + mean_{wL,F}) \tag{5.5}$$

and

$$\mathbf{M}_{CR,h}^{g+1} = 0.5(\mathbf{M}_{CR,h}^{g} + mean_{wL,CR}) \tag{5.6}$$

5.2.3 Further Improvements to the SHADE Algorithms

There have been many improvements to the SHADE algorithm [89, 90, 87, 91]. In this section, only the linear decrease in population and the distance-based parameter adaptation mechanisms are discussed.

Linear Decrease in Population Size

Linear population reduction was introduced to SHADE in order to enhance exploitation in later stages of the evolution. The resulting algorithm from this update is LSHADE [87]. At every generation, LSHADE calculates the new population by using

$$\mathrm{NP}_{g+1} = \mathrm{round}\left(\frac{E(\mathrm{NP}_{\min} - \mathrm{NP}_{\max})}{E_{\max}} + \mathrm{NP}_{\max}\right), \tag{5.7}$$

where $\mathrm{NP}_{\min}$ and $\mathrm{NP}_{\max}$ are the minimum and initial population sizes respectively, while E and $E_{\max}$ are the current and maximum number of objective function evaluations. When the new population size is lower than the previous value, the worst individuals are removed from the population.

Distance-based Parameter Adaptation Mechanism

The distance-based parameter adaptation mechanism, proposed in [47], was motivated by the observation that the original adaptation mechanism for scaling factor and crossover rate values promotes exploitation over exploration. According to [47], this phenomenon can lead to premature convergence, which could be a problem, especially in higher dimensions. To prevent this problem, they proposed a Euclidean distance-based mechanism, where CR and F values connected with the individual that moved the farthest will have the highest weight and H is the problem dimension:

$$w_k = \frac{\sqrt{\sum_{j=1}^{H}(u_{k,j,g} - x_{k,j,g})^2}}{\sum_{m=1}^{|S_{CR}|}\sqrt{\sum_{m=1}^{H}(u_{m,j,g} - x_{m,j,g})^2}}. \tag{5.8}$$

Algorithm 5: SHADE Algorithm [47]

Input: $\mathrm{NP}_{\mathrm{init}}, \mathrm{Hm}$, and termination criteria;
Output: x_{best} as the best found solution;
$\mathrm{NP} = \mathrm{NP}_{\mathrm{init}}, G = 0, \mathbf{x}_{\mathrm{best}} = \{\}, k = 1, p_{min} = 2/\mathrm{NP}, A = \emptyset$;
Randomly generate population $P = (\mathbf{x}_{1,G}, ..., \mathbf{x}_{NP,G})$;
Set $\mathbf{M}_F$ and $\mathbf{M}_{CR}$, $P_{\mathrm{new}} = \{\}$, $\mathbf{x}_{\mathrm{best}}$ = best from population P;
while *stopping criteria not met* **do**
    $\mathbf{S}_F = \emptyset$, $\mathbf{S}_{CR} = \emptyset$;
    for $i \leftarrow 1$, NP **do**
        Set F_i by (5.2) and CR_i by (5.3), $\mathbf{x}_{i,G} = P[i]$, $p_i = U[p_{\mathrm{min}}, 0.2]$;
        $\mathbf{v}_{i,G}$ by mutation (2.3), $\mathbf{u}_{i,G}$ by crossover (2.4);
        if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ **then**
            $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G}$, $A \leftarrow \mathbf{x}_{i,G}$, $S_F \leftarrow F_i$, $S_{CR} \leftarrow CR_i$;
            else
            $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$;
        if $|A| > NP$ **then**
            Randomly delete $|A| - \mathrm{NP}$ individuals from A;
        $P_{\mathrm{new}} \leftarrow \mathbf{x}_{i,G+1}$
    Calculate $\mathrm{NP}_{\mathrm{new}}$ according to (5.7);
    if $NP_{new} < NP$ **then**
        Sort individuals in P according to their objective function values and remove NP - $\mathrm{NP}_{\mathrm{new}}$ worst ones;
        $\mathrm{NP} = \mathrm{NP}_{\mathrm{new}}$;
    if $\boldsymbol{S}_F \neq \emptyset$, $\boldsymbol{S}_{CR} \neq \emptyset$ **then**
        Update $\mathbf{M}_{F,k}$ and $\mathbf{M}_{CR,k}$ with Lehmer mean computed by (5.4) and distance-based weights in (5.8), k++;
    if $k > Hm$ **then**
        $k = 1$;
    $P = P_{\mathrm{new}}, P_{\mathrm{new}} = \{\}$, $\mathbf{x}_{best}$ = best from population P;
    $G \leftarrow G + 1$

The code for the SHADE variant is available online (https://github.com/wikkiw/DISH). Algorithm 5 depicts the pseudo-code of the SHADE optimization process. Specifically, on line 24, the DISH mechanism presented in equation (5.8) is invoked.

The main drawback of this SHADE variant is a slightly higher computational complexity for calculating the weights for historical memory updates. However, time complexity measurements in [47] have shown that this does not impact the overall computation time. In addition, the distance method is based on the Euclidean distance between the trial and the original individual which slightly increases the complexity of the weight calculation, nevertheless, the overall complexity of the algorithm is similar.

5.3 Simulation Environment

The simulations are performed using the FogWorkflowSim simulator and are performed on a computer with 64-bit Windows 10 operating system, Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz and 16 GB RAM. The population size is set to 50 for each algorithm. The canonical DE crossover probability is 0.4 and the scaling factor is 1.2. The number of iterations is 100. The algorithms are evaluated using the five scientific workflows previously described, with three different task amounts per workflow. The simulations are performed 10 times for each workflow to get the average performance of the algorithms. The number of cloud VMs, fog VMs and end devices are 5, 5 and 1, respectively. The computational capacity of a device or VM is measured in million instructions per second (MIPS). The processing capacity of the end device is 1000 MIPS, each fog VM is 1300 MIPS and that of a cloud VM is 1600 MIPS. The task execution cost is 0.48 and 0.96 for the fog and cloud VMs respectively.

5.4 Simulation Results and Discussion

In Figures 5.1-5.3, the results are shown for makespan, cost and energy consumption for the Montage workflow. The results show that the canonical DE is exhibiting poorer performance compared to the SHADE variant across the three performance metrics and task amounts.

The better performance exhibited by SHADE is likely due to the distance-based parameter adapta-

tion mechanism which enables a better exploratory ability and avoidance of premature convergence. As expected, all the performance metric values are higher as the workflow task amount increases. The metrics for 20 tasks is very low but it rises sharply when the number of tasks increases to 100. Specifically, in Fig. 5.2, cost increases from less than $100 for 20 tasks to beyond $200 at 100 tasks. In Fig. 5.3, energy increases from less than 600J for 20 tasks to beyond 1200J at 100 tasks.

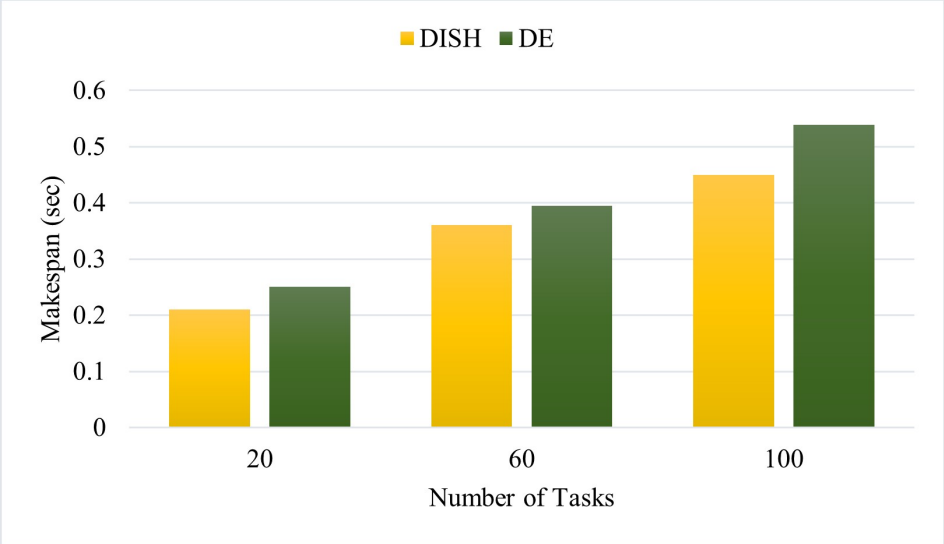


Figure 5.1: Makespan for Montage

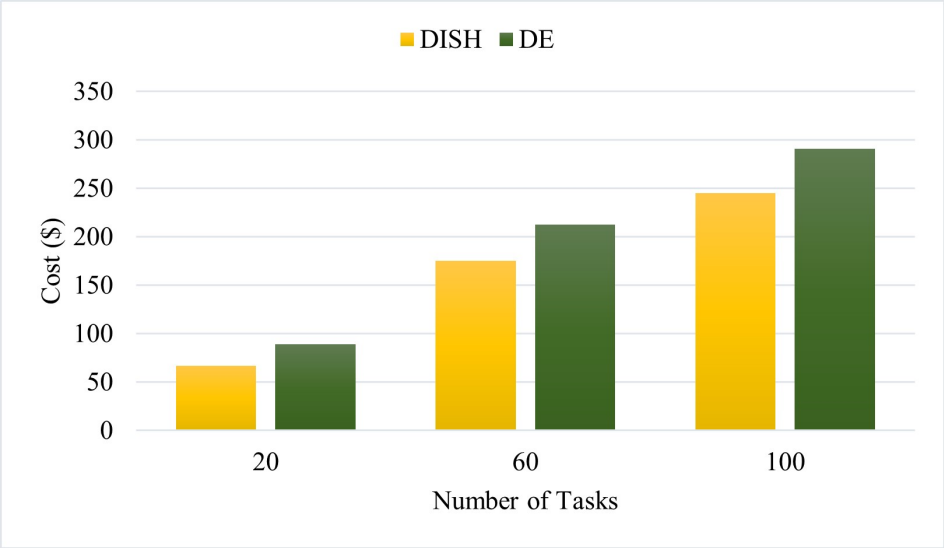


Figure 5.2: Cost for Montage

In Figures 5.4-5.9, the results are shown for makespan, cost and energy consumption for the CyberShake and Epigenomics workflow, respectively. The results are similar to what has been

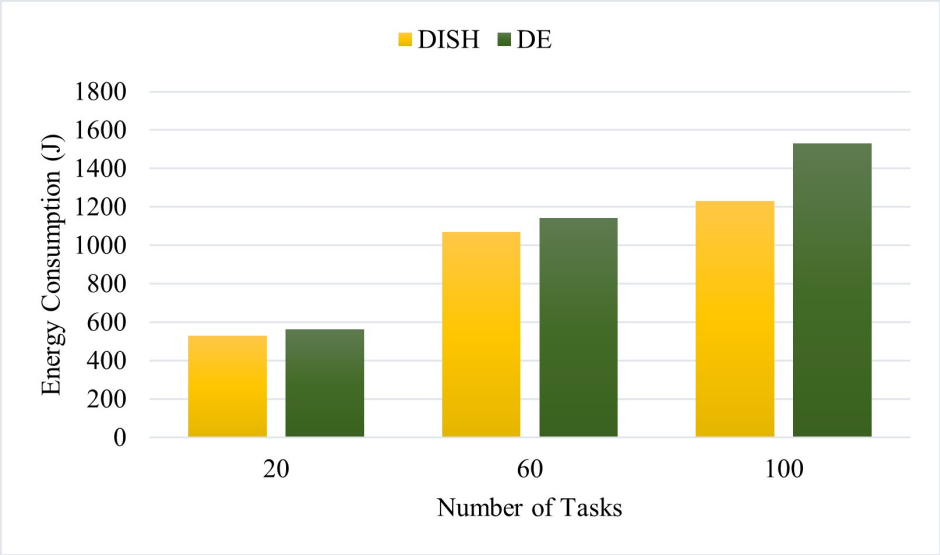


Figure 5.3: Energy consumption for Montage

observed for the Montage workflow, whereby the canonical DE is exhibiting poorer performance compared to DISH. However, the metric values obtained for CyberShake and Epigenomics are much higher. This is due to the higher workflow task sizes and runtimes compared to Montage.

On the other hand, in Fig. 5.5, the cost decreases from over $30 000 for 30 tasks to less than $15 000 at 100 tasks. The cost goes down considerably as the number of tasks increases. It looks like, for CyberShake in particular, the DE and DISH algorithms allocate more tasks to the end device as the number of tasks increases thereby reducing the cost. There are low costs due to less usage of cloud and fog servers. This leads to high energy consumption due to increased usage of the end device as shown by Fig. 5.6, where the energy consumption increases from less than 250 000J for 30 tasks to over 500 000J at 100 tasks.

In Figs. 5.7-5.9, similar to what was observed in Chapter 3, the metrics for Epigenomics on 24 and 47 tasks is very low but it rises significantly when the number of tasks increases to 100. The map jobs in the Epigenomics workflow [1] become significantly more computationally intensive as the number of tasks increases. Makespan increases from less than 10 sec for 24 and 47 tasks to beyond 50 sec at 100 tasks; cost increases from less than $5 000 for 24 and 47 tasks to beyond $40 000 at 100 tasks; energy increases from less than 50 000J for 24 and 47 tasks to beyond 250 000J at 100 tasks. The DISH algorithm also displays improved performance as the workflow size increases. It outperforms the canonical DE by nearly 20% for Epigenomics, at 100 tasks, for the

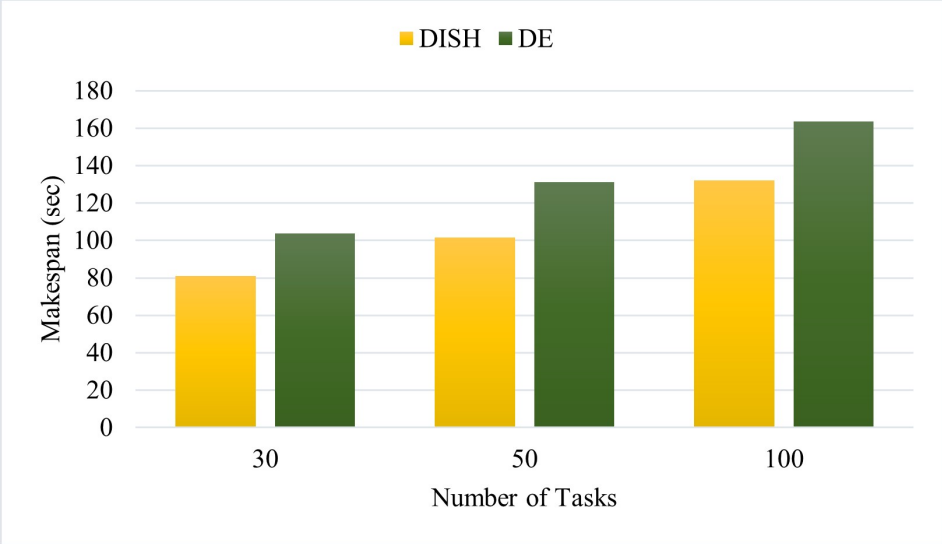


Figure 5.4: Makespan for CyberShake

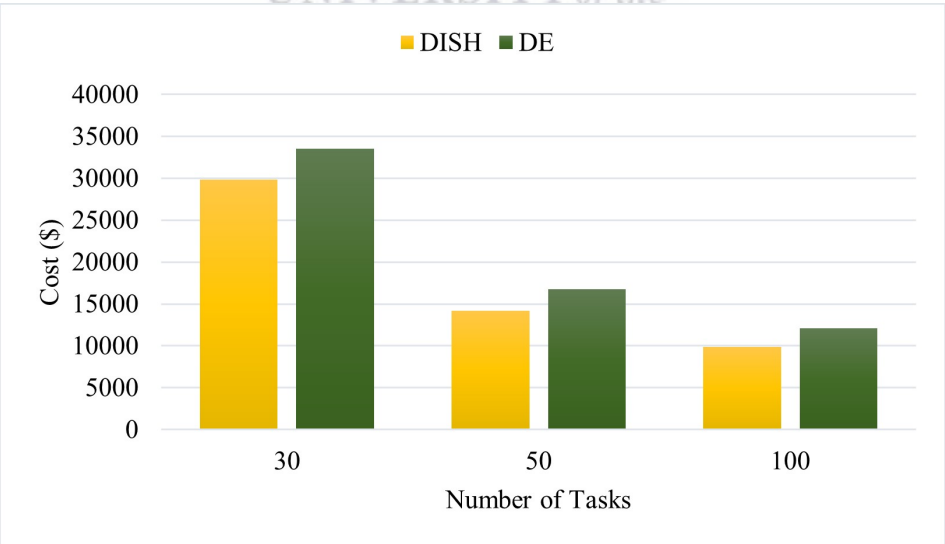


Figure 5.5: Cost for CyberShake

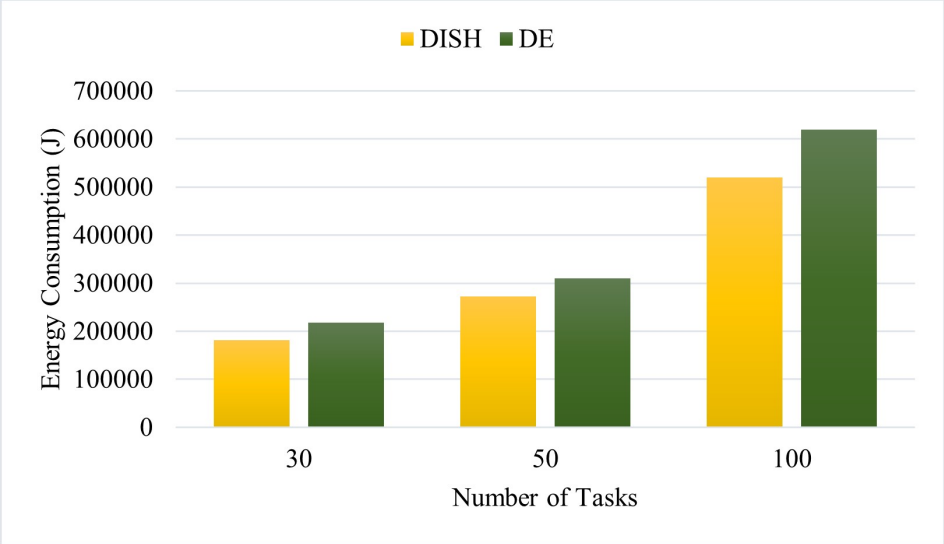


Figure 5.6: Energy consumption for CyberShake

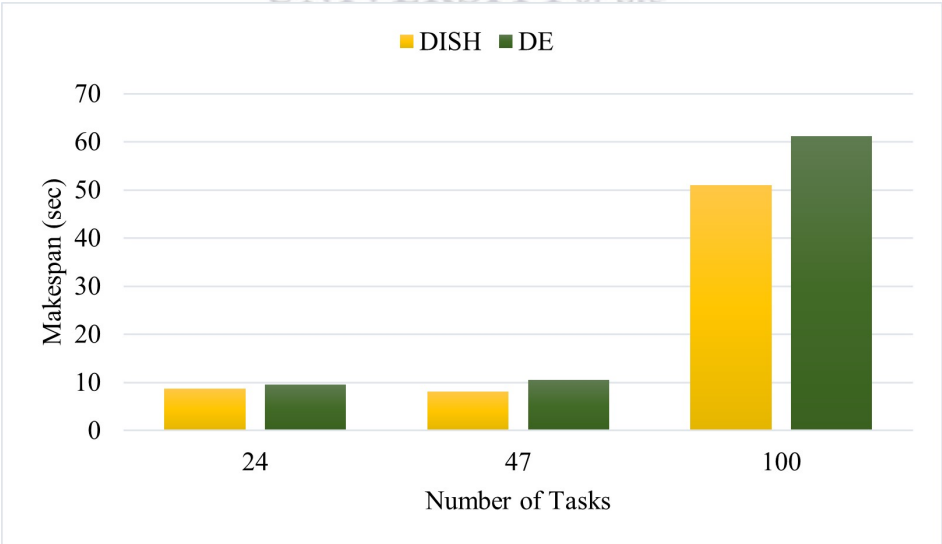


Figure 5.7: Makespan for Epigenomics

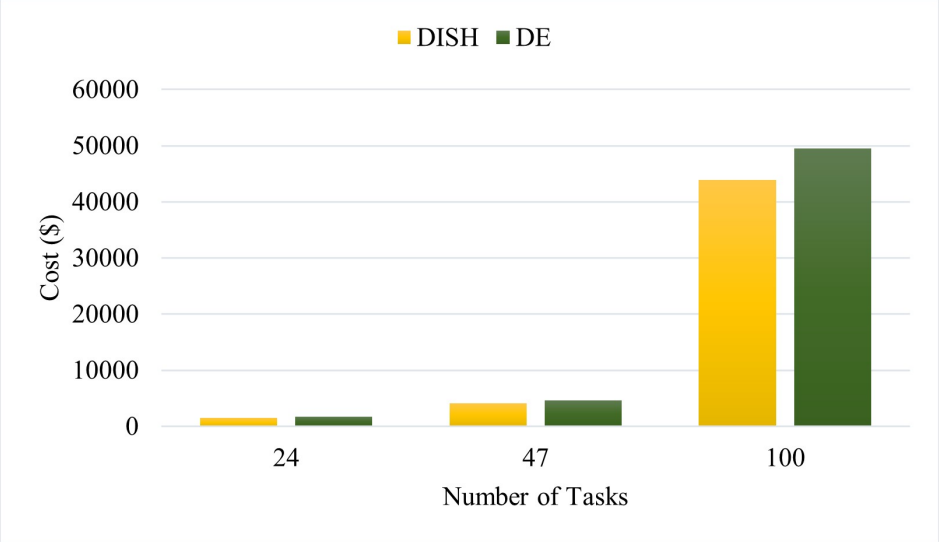


Figure 5.8: Cost for Epigenomics

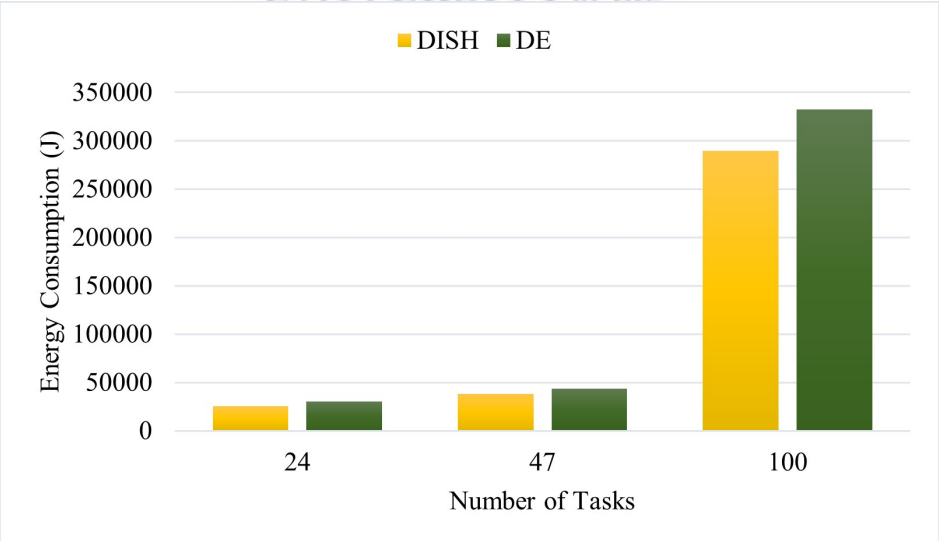


Figure 5.9: Energy consumption for Epigenomics

three performance metrics. This is due to the better exploratory capability of DISH.

In Figures 5.10-5.15, the results are shown for makespan, cost and energy consumption for the LIGO workflow. Similarly to the observations for the other workflows, the results show that the canonical DE is exhibiting poorer performance compared to the DISH algorithm. The performance metric values are higher as the workflow task amount increases. The metrics for 30 tasks is low but it rises considerably when the number of tasks increases to 100.

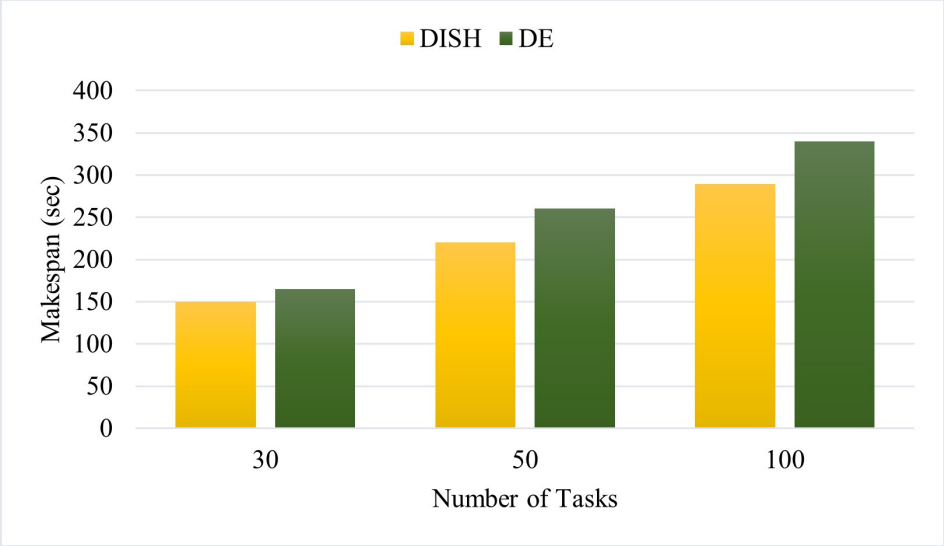


Figure 5.10: Makespan for LIGO

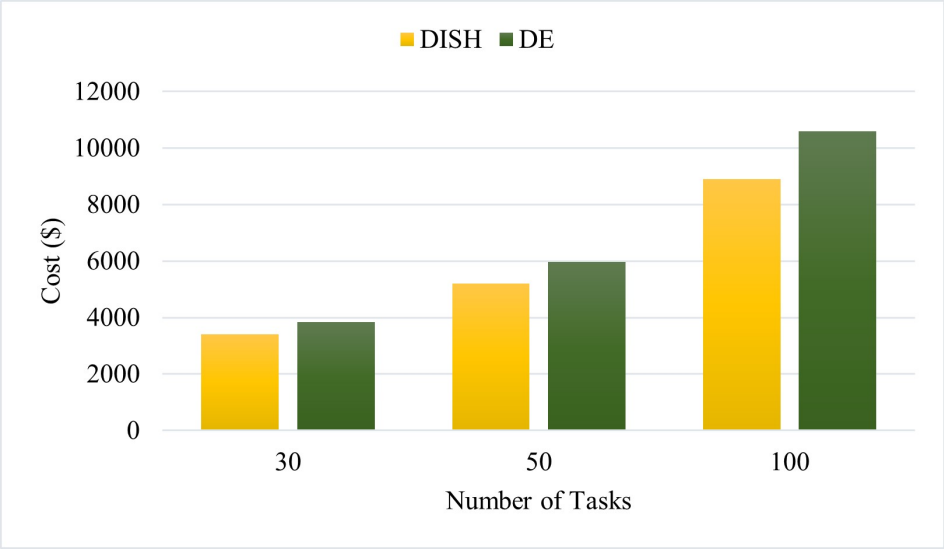


Figure 5.11: Cost for LIGO

Specifically for the LIGO workflow, in Fig. 5.10, makespan increases from less than 200 sec for 30

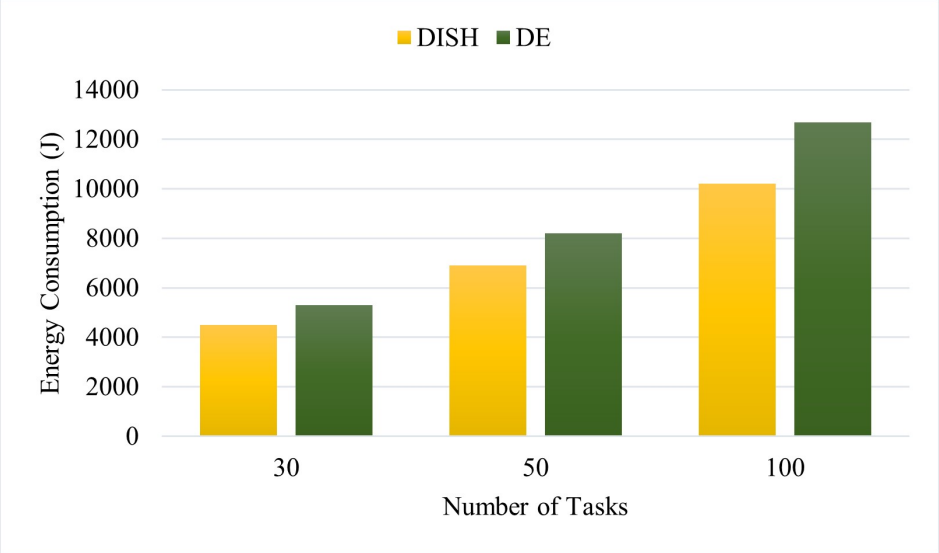


Figure 5.12: Energy consumption for LIGO

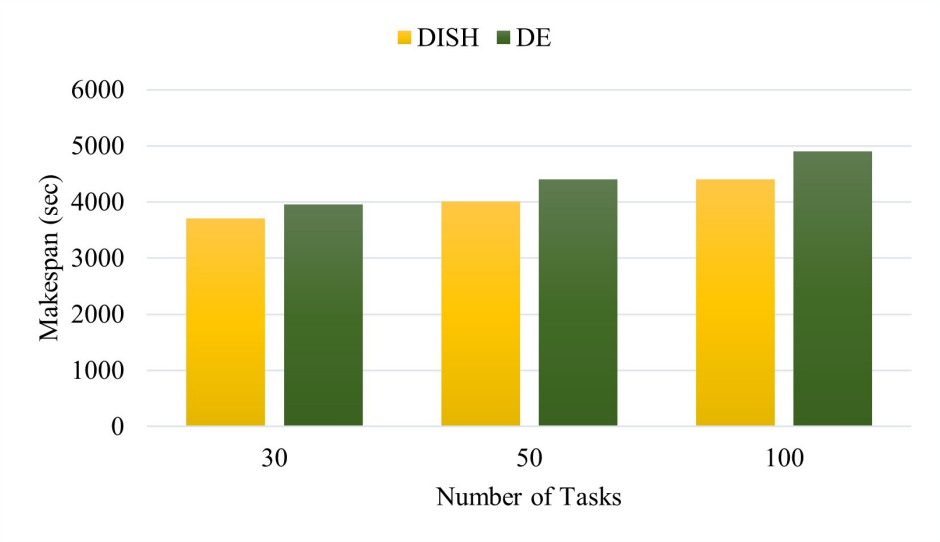


Figure 5.13: Makespan for SIPHT

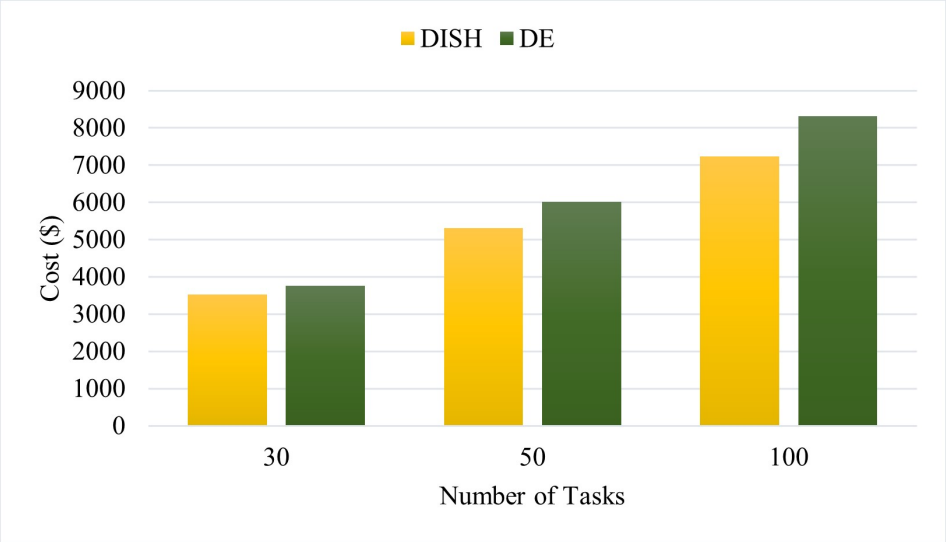


Figure 5.14: Cost for SIPHT

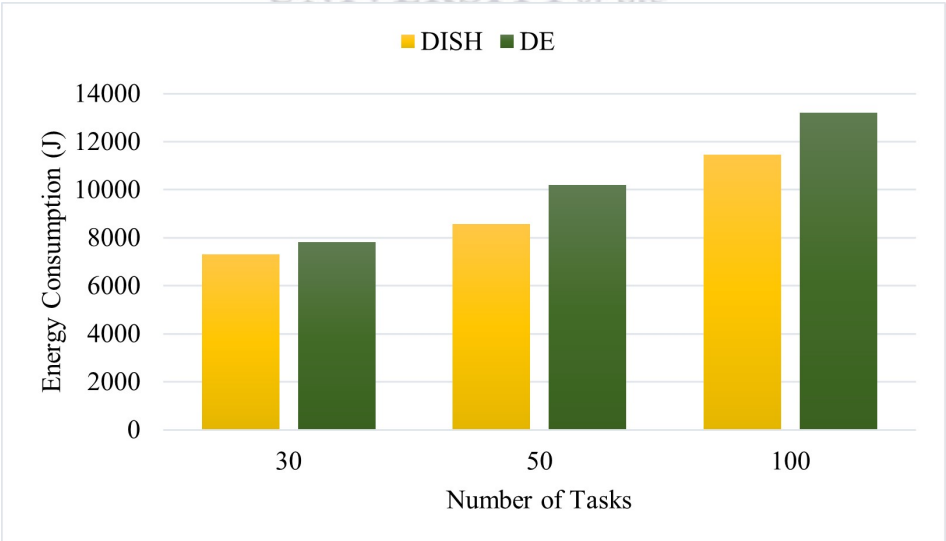


Figure 5.15: Energy consumption for SIPHT

tasks to beyond 250 sec at 100 tasks. In Fig. 5.11, cost increases from less than $4000 for 30 tasks to beyond $8000 at 100 tasks. In Fig. 5.12, energy consumption increases from less than 6000J for 30 tasks to over 10 000J at 100 tasks.

Figures 5.13-5.15 show the results for makespan, cost and energy consumption for the SIPHT workflow. In Fig. 5.14, cost increases from less than $4000 for 30 tasks to beyond $7000 at 100 tasks. In Fig. 5.15, energy consumption increases from less than 8000J for 30 tasks to over 10 000J at 100 tasks.

Overall, the results indicate that the performance of the DISH is better than that of the canonical DE. The DISH algorithm also seems to show better performance as the workflow task amount increases, hence the DISH algorithm may be more suitable for large-scale workflow scheduling where the solution vector is of a high dimension.

5.5 Chapter Summary

This chapter investigated the application of the canonical DE and a DE variant, known as SHADE, to the problem of workflow scheduling in cloud-fog environments. Simulations are conducted on the five well-known scientific workflows with different sizes. The results obtained indicate that the SHADE variant outperformed the canonical DE for all workflow instances and it has the potential to improve workflow processing efficiency and reduce energy consumption. The SHADE variant also displayed better performance than the canonical DE, especially for the workflow instances with higher task amounts. The next chapter summarizes and concludes this study. The possible future work is also presented.

Chapter 6

Conclusion and Future Works

6.1 Introduction

This Chapter presents the concluding remarks as well as recommendations for future work. It is organised as follows. Section 6.1 presents concluding remarks, while section 6.2 presents future perspectives.

6.2 Conclusion

The main aim of this work was to develop population-based multi-objective algorithms for scheduling scientific workflows in the emerging three-tier architecture, that incorporates end-devices, fog devices and cloud-devices. Unlike the traditional two-tier cloud infrastructure, the cloud-fog paradigm helps to reduce processing latency; this is an excellent attribute for the execution of complex scientific workflows.

The work begins by presenting a comprehensive literature review focusing on the following issues: the basics of scientific workflows, cloud-fog environments; the theoretical foundations for population-based algorithms such as Particle Swarm Optimisation (PSO), Genetic Algorithm (GA), and Differential Evolution (DE); the applications of population-based algorithms to scientific workflows; and simulation frameworks. The gaps, identified in the literature review, set the platform for the contributions emanating from this work.

Pegasus workflows, such as Montage, Cybershake, Epigenomics, LIGO and SIPHT, were used in this work. The FogWorkflowSim simulator [28] was used as the platform for implementing and testing the population-based algorithms for scientific workflow scheduling throughout this work. The results of this work can divided in three phases.

The first phase started with the development of a weighted sum-based objective function for workflow scheduling, by combining makespan, computation and communication costs, and energy consumed in active as well as in idle mode on all the computation devices; this helped to ensure that a holistic view of energy consumption is incorporated in the optimization process. The Differential Evolution (DE) algorithm was introduced to this simulator for the very first time, courtesy of this work. Two sets of experiments were conducted.

In the first set, a comprehensive evaluation of PSO, GA, DE and GA-PSO on the workflow scheduling problem in the three-tier cloud-fog environment, was conducted. Makespan, cost and energy were used as objectives to evaluate the quality of the solutions. The results show that the GA-PSO algorithm performed slightly better than the standard approaches. Although the number of tasks per workflow was limited to 100, this performance showed the potential of hybrid algorithms that synergistically combine the good attributes of the standard algorithms to improve performance. In the second set of experiments, a performance evaluation of the canonical PSO algorithm, applied to both the traditional cloud and the emerging three-tier cloud-fog environment, was conducted. The cloud-fog environment yielded lower makespan, cost, and energy consumption. This clearly shows the efficacy of the advantages of running scientific workflows in cloud-fog environments.

In the second phase, the objective function was further expanded to incorporate load balancing on the fog as well as on the cloud resources to ensure workloads are distributed evenly for more efficient processing of workflows. A brief survey on the state-of-the Multi-Swarm PSO algorithms and techniques was implemented. Then the work proceeded to develop a Multi-Swarm based PSO algorithm to address the issue of premature convergence with the canonical PSO. The proposed MS-PSO outperforms the canonical PSO by atleast 10% on all scientific workflows and under all performance metrics. It also competes fairly well against the other approaches and it is more stable and reliable. This provides insight into how the improvements of the standard algorithms, that retain the good attributes, can improve the scheduling of workflows. The number of task per workflow was increased to 500.

In the third phase, this research applied a modern DE variant known as SHADE to workflow scheduling and compared it with the canonical DE. The SHADE variant outperformed the canonical DE by atleast 10%, for each performance metric, on all scientific workflow instances. The SHADE variant also displayed better performance than the canonical DE for the large workflow instances, indicating that it may be more suitable for large-scale workflow scheduling.

6.3 Future Works

In terms of future work, the following can be pursued:

- The number of tasks in each of the scientific workflows can be increased to over 1000, specifically the evaluation of large-scale scientific workflows. This will require more computing resources.
- The optimization objectives can be expanded to include reliability and fault tolerance, and the incorporation of deadline and budget constraints.
- Development of a multiple species population-based algorithm, incorporating population update mechanisms from several algorithmic frameworks (MS-PSO, DE, GA), for scientific workflow scheduling. This will incorporate the positive attributes of all algorithms.
- A framework that hybridizes population-based scheduling mechanisms with concepts drawn from dynamic scientific workflow scheduling [92] and multi-objective reinforcement learning-based scientific workflow scheduling [93] can also be investigated.

References

[1] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *2008 third workshop on workflows in support of large-scale science*. IEEE, 2008, pp. 1–10.

[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[3] J. Gantz, D. Reinsel *et al.*, "Extracting value from chaos," *IDC iview*, vol. 1142, no. 2011, pp. 1–12, 2011.

[4] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for marketoriented cloud computing," *Proceedings of the 1st International Conference on Cloud Computing*, vol. 5931, pp. 24–44, 2009.

[5] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (iaas)," *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60–63, 2010.

[6] Y. Zhao, Y. Li, I. Raicu, S. Lu, C. Lin, Y. Zhang, W. Tian, and R. Xue, "A service framework for scientific workflow management in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 930–944, 2014.

[7] W. Song, F. Chen, H.-A. Jacobsen, X. Xia, C. Ye, and X. Ma, "Scientific workflow mining in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2979–2992, 2017.

[8] T. Warnow, *Bioinformatics and phylogenetics: seminal contributions of Bernard Moret*. Springer, 2019, vol. 29.

[9] S. Jha, S. Lathrop, J. Nabrzyski, and L. Ramakrishnan, "Incorporating scientific workflows in computing research processes," *Computing in Science & Engineering*, vol. 21, no. 4, pp. 4–6, 2019.

[10] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *Journal of grid computing*, vol. 3, no. 3, pp. 171–200, 2005.

[11] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1–19, 2017.

[12] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Generation Computer Systems*, vol. 97, pp. 361–378, 2019.

[13] A. Singh *et al.*, "A multi-objective workflow scheduling algorithm for cloud environment," in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. IEEE, 2018, pp. 1–6.

[14] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid ga-pso algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[15] M. Farid, R. Latip, M. Hussin, and N. A. W. Abdul Hamid, "A survey on qos requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, p. 551, 2020.

[16] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *big data and cognitive computing*, vol. 2, no. 2, p. 10, 2018.

[17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[18] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, 2017.

[19] V. De Maio and D. Kimovski, "Multi-objective scheduling of extreme data scientific workflows in fog," *Future Generation Computer Systems*, vol. 106, pp. 171–184, 2020.

[20] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of pso-based scheduling algorithms in cloud computing," *Journal of Network and Systems Management*, vol. 25, no. 1, pp. 122–158, 2017.

[21] J. Lu, J. Zhang, and J. Sheng, "Enhanced multi-swarm cooperative particle swarm optimizer," *Swarm and Evolutionary Computation*, vol. 69, p. 100989, 2022.

[22] Y.-L. Hu and L. Chen, "A nonlinear hybrid wind speed forecasting model using lstm network, hysteretic elm and differential evolution algorithm," *Energy conversion and management*, vol. 173, pp. 123–142, 2018.

[23] A. Ghosh, N. D. Jana, S. Mallik, and Z. Zhao, "Designing optimal convolutional neural network architecture using differential evolution algorithm," *Patterns*, vol. 3, no. 9, p. 100567, 2022.

[24] S. Li, W. Gong, C. Hu, X. Yan, L. Wang, and Q. Gu, "Adaptive constraint differential evolution for optimal power flow," *Energy*, vol. 235, p. 121362, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544221016108

[25] S. Gao, K. Wang, S. Tao, T. Jin, H. Dai, and J. Cheng, "A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models," *Energy Conversion and Management*, vol. 230, p. 113784, 2021.

[26] V. Parque and T. Miyashita, "Smooth curve fitting of mobile robot trajectories using differential evolution," *IEEE Access*, vol. 8, pp. 82 855–82 866, 2020.

[27] G. Luo, L. Zou, Z. Wang, C. Lv, J. Ou, and Y. Huang, "A novel kinematic parameters calibration method for industrial robot based on levenberg-marquardt and differential evolution hybrid algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102165, 2021.

[28] X. Liu, L. Fan, J. Xu, X. Li, L. Gong, J. Grundy, and Y. Yang, "Fogworkflowsim: An automated simulation toolkit for workflow performance evaluation in fog computing," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 1114–1117.

[29] The Pegasus website, [Online]. Available: https://pegasus.isi.edu/. [Accessed 14 September 2022].

[30] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci*, vol. 8, no. 4, pp. 1–10, 2016.

[31] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*. IEEE, 2015, pp. 73–78.

[32] C.-W. Tsai and J. J. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Systems Journal*, vol. 8, no. 1, pp. 279–291, 2013.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization in: Neural networks," in *Proceedings IEEE International Conference on 1995*, 1942, pp. 1942–1948.

[34] F. Wang, H. Zhang, and A. Zhou, "A particle swarm optimization algorithm for mixed-variable optimization problems," *Swarm and Evolutionary Computation*, vol. 60, p. 100808, 2021.

[35] H. J. Na and S.-J. Yoo, "Pso-based dynamic uav positioning algorithm for sensing information acquisition in wireless sensor networks," *IEEE Access*, vol. 7, pp. 77 499–77 513, 2019.

[36] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial intelligence*, vol. 40, no. 1-3, pp. 235–282, 1989.

[37] H. C. Lau, T. Chan, W. Tsui, and W. Pang, "Application of genetic algorithms to solve the multidepot vehicle routing problem," *IEEE transactions on automation science and engineering*, vol. 7, no. 2, pp. 383–392, 2009.

[38] G. Verma and V. Verma, "Role and applications of genetic algorithm in data mining," *International journal of computer applications*, vol. 48, no. 17, pp. 5–8, 2012.

[39] S. Wei, B. Wei, Y. Chen, L. Hao, Y. Huang, W. Dai, and B. Liang, "Time-dependent short-term observational scheduling method for yunnan 40 m radio telescope using a genetic algorithm," *Astrophysics and Space Science*, vol. 367, no. 9, pp. 1–8, 2022.

[40] R. B. Kasat, A. K. Ray, and S. K. Gupta, "Applications of genetic algorithm in polymer science and engineering," *Materials and Manufacturing Processes*, vol. 18, no. 3, pp. 523–532, 2003.

[41] P. W. Khan and Y.-C. Byun, "Genetic algorithm based optimized feature engineering and hybrid machine learning for effective energy consumption prediction," *IEEE Access*, vol. 8, pp. 196 274–196 286, 2020.

[42] I. Gupta, S. Gupta, A. Choudhary, and P. K. Jana, "A hybrid meta-heuristic approach for load balanced workflow scheduling in iaas cloud," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2019, pp. 73–89.

[43] A. Awad, N. El-Hefnawy, and H. Abdel_kader, "Enhanced particle swarm optimization for task scheduling in cloud computing environments," *Procedia Computer Science*, vol. 65, pp. 920–929, 2015.

[44] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[45] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation*, vol. 10, no. 6, pp. 646–657, 2006.

[46] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and evolutionary computation*, vol. 27, pp. 1–30, 2016.

[47] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda, "Distance based parameter adaptation for success-history based differential evolution," *Swarm and Evolutionary Computation*, vol. 50, p. 100462, 2019.

[48] Q. Pan, J. Tang, H. Wang, H. Li, X. Chen, and S. Lao, "Sfsade: an improved self-adaptive differential evolution algorithm with a shuffled frog-leaping strategy," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3937–3978, 2022.

[49] L. Karthikeyan, C. Vijayakumaran, S. Chitra, and S. Arumugam, "Saldeft: Self-adaptive learning differential evolution based optimal physical machine selection for fault tolerance problem in cloud," *Wireless Personal Communications*, vol. 118, no. 2, pp. 1453–1480, 2021.

[50] C. Cubukcuoglu, B. Ekici, M. F. Tasgetiren, and S. Sariyildiz, "Optimus: self-adaptive differential evolution with ensemble of mutation strategies for grasshopper algorithmic modeling," *Algorithms*, vol. 12, no. 7, p. 141, 2019.

[51] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE transactions on cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2019.

[52] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, "Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review," *Journal of Grid Computing*, vol. 18, no. 3, pp. 327–356, 2020.

[53] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, 2013.

[54] M. Adhikari and T. Amgoth, "Multi-objective accelerated particle swarm optimization technique for scientific workflows in iaas cloud," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 1448–1454.

[55] A. Ghorbannia Delavar and Y. Aryan, "Hsga: a hybrid heuristic algorithm for workflow scheduling in cloud systems," *Cluster computing*, vol. 17, no. 1, pp. 129–137, 2014.

[56] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A gsa based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Generation Computer Systems*, vol. 83, pp. 14–26, 2018.

[57] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.

[58] M. S. Kumar, I. Gupta, S. K. Panda, and P. K. Jana, "Granularity-based workflow scheduling algorithm for cloud computing," *The Journal of Supercomputing*, vol. 73, no. 12, pp. 5440–5464, 2017.

[59] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[60] E. Ilavarasan and P. Thambidurai, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *Journal of Computer sciences*, vol. 3, no. 2, pp. 94–103, 2007.

[61] S. J. Nirmala and S. Bhanu, "Catfish-pso based scheduling of scientific workflows in iaas cloud," *Computing*, vol. 98, no. 11, pp. 1091–1109, 2016.

[62] X. Wang, B. Cao, C. Hou, L. Xiong, and J. Fan, "Scheduling budget constrained cloud workflows with particle swarm optimization," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2015, pp. 219–226.

[63] A. Verma and S. Kaushal, "Cost-time efficient scheduling plan for executing workflows in the cloud," *Journal of Grid Computing*, vol. 13, no. 4, pp. 495–506, 2015.

[64] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control," *Journal of grid computing*, vol. 11, no. 4, pp. 633–651, 2013.

[65] I. Casas, J. Taheri, R. Ranjan, and A. Y. Zomaya, "Pso-ds: a scheduling engine for scientific workflow managers," *The Journal of Supercomputing*, vol. 73, no. 9, pp. 3924–3947, 2017.

[66] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *Journal of Network and Computer Applications*, vol. 114, pp. 108–122, 2018.

[67] G. Ismayilov and H. R. Topcuoglu, "Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing," *Future Generation computer systems*, vol. 102, pp. 307–322, 2020.

[68] M. Adhikari and T. Amgoth, "An intelligent water drops-based workflow scheduling for iaas cloud," *Applied Soft Computing*, vol. 77, pp. 547–566, 2019.

[69] D. A. Monge, E. Pacini, C. Mateos, E. Alba, and C. G. Garino, "Cmi: An online multi-objective genetic autoscaler for scientific and engineering workflows in cloud infrastructures with unreliable virtual machines," *Journal of Network and Computer Applications*, vol. 149, p. 102464, 2020.

[70] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, qos-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing dvfs and approximate computations," *Future Generation Computer Systems*, vol. 96, pp. 216–226, 2019.

[71] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," *8th IEEE International Conference on E-science*, pp. 1–8, 2012.

[72] S. V. Margariti, V. V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective," *Future Internet*, vol. 12, no. 5, p. 89, 2020.

[73] M. Gill and D. Singh, "A comprehensive study of simulation frameworks and research directions in fog computing," *Computer Science Review*, vol. 40, p. 100391, 2021.

[74] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[75] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[76] J.-J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* IEEE, 2005, pp. 124–129.

[77] I. Bugajski, P. Listkiewicz, A. Byrski, M. Kisiel-Dorohinicki, W. Korczynski, T. Lenaerts, D. Samson, B. Indurkhya, and A. Nowé, "Enhancing particle swarm optimization with socio-cognitive inspirations," *Procedia Computer Science*, vol. 80, pp. 804–813, 2016.

[78] S. Wang, G. Liu, M. Gao, S. Cao, A. Guo, and J. Wang, "Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators," *Information Sciences*, vol. 540, pp. 175–201, 2020.

[79] F. T. Varna and P. Husbands, "Hidms-pso: A new heterogeneous improved dynamic multi-swarm pso algorithm," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 473–480.

[80] X. Xia, Y. Tang, B. Wei, Y. Zhang, L. Gui, and X. Li, "Dynamic multi-swarm global particle swarm optimization," *Computing*, vol. 102, no. 7, pp. 1587–1626, 2020.

[81] B. Wei, Y. Tang, X. Jin, M. Jiang, Z. Ding, and Y. Huang, "A dynamic multi-swarm particle swarm optimization with global detection mechanism," *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, vol. 15, no. 4, pp. 1–23, 2021.

[82] X. Jiang, Y. Yue, Y. Min, Q. Zhang, and X. Gui, "Particle swarm optimization with multiple adaptive subswarms," in *Journal of Physics: Conference Series*, vol. 1757, no. 1. IOP Publishing, 2021, p. 012024.

[83] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE transactions on evolutionary computation*, vol. 10, no. 3, pp. 281–295, 2006.

[84] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 225–239, 2004.

[85] J. Guzmán, M. García-Valdez, and J. J. Merelo-Guervós, "Can communication topology improve a multi-swarm pso algorithms?" in *Workshop on Engineering Applications*. Springer, 2021, pp. 3–12.

[86] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 71–78.

[87] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 1658–1665.

[88] P. S. Bullen, *Handbook of means and their inequalities*. Springer Science & Business Media, 2013, vol. 560.

[89] S. Chakraborty, S. Sharma, A. K. Saha, and S. Chakraborty, "Shade–woa: A metaheuristic algorithm for global optimization," *Applied Soft Computing*, vol. 113, p. 107866, 2021.

[90] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE congress on evolutionary computation (CEC)*. IEEE, 2017, pp. 1311–1318.

[91] X. Wang, C. Li, J. Zhu, and Q. Meng, "L-shade-e: Ensemble of two differential evolution algorithms originating from l-shade," *Information Sciences*, vol. 552, pp. 201–219, 2021.

[92] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 2–18, 2015.

[93] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "A multi-objective reinforcement learning algorithm for deadline constrained scientific workflow scheduling in clouds," *Frontiers of Computer Science*, vol. 15, no. 5, pp. 1–12, 2021.